Exhibit E-1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

EXPRESS MAIL NUMBER: EV 978 428 202 US DATE OF DEPOSIT: February 8, 2008

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage Via Express Mail No. EV 978 428 202 US in an envelope addressed to: Central Reexamination Unit (CRU), ATTN: "Box Inter Partes Reexam", Commissioner for Patents, United States Patent and Trademark Office, P.O. Box 1450, Alexandria, VA. 22313-1450.

Rosa A. Caviedes

Rosa A. Caviedes

* * *

CERTIFICATE OF MAILING BY EXPRESS MAIL

COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, VA 22313-14500

Sir:

Transmitted herewith for filing are the following:

1. Transmittal Form;
2. Response to Notice of Failure to Comply with Inter Partes Reexamination Request Filing Requirements (37 CFR 1.915(D);
3. Replacement Attachment to Request for Inter Partes Reexamination of U.S. Patent No. 6,857,001;
4. Certificate of Mailing By Express Mail No.: EV 978 428 202 US; and
5. Return Post Card.

EV978428202US

PTO/SB/21 (10-07)
Approved for use through 10/31/2007. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Total Number of Pages in This Submission	186

U.S. Patent No.	6,857,001
Filing Date	June 7, 2002
First Named Inventor	Hitz et al.
Art Unit	N/A
Examiner Name	N/A
Attorney Docket Number	347155-29

ENCLOSURES *(Check all that apply)*

- ☐ Fee Transmittal Form
  - ☐ Fee Attached
- ☐ Amendment/Reply
  - ☐ After Final
  - ☐ Affidavits/declaration(s)
- ☐ Extension of Time Request
- ☐ Express Abandonment Request
- ☐ Information Disclosure Statement & PTO-1449 with 13 references
- ☐ Certified Copy of Priority Document(s)
- ☐ Reply to Missing Parts/ Incomplete Application
  - ☐ Reply to Missing Parts under 37 CFR 1.52 or 1.53
- ☐ Drawing(s)
- ☐ Licensing-related Papers
- ☐ Petition
- ☐ Petition to Convert to a Provisional Application
- ☐ Power of Attorney, Revocation Change of Correspondence Address
- ☐ Terminal Disclaimer
- ☒ Response to Notice of Failure to Comply with Inter Partes Re-Examination Request Filing Requirements (37 CFR 1.915(D))
- ☐ CD, Number of CD(s) _____
  - ☐ Landscape Table on CD
- ☐ After Allowance Communication to TC
- ☐ Appeal Communication to Board of Appeals and Interferences
- ☐ Appeal Communication to TC **(Appeal Notice, Brief, Reply Brief)**
- ☐ Proprietary Information
- ☐ Status Letter
- ☒ Other Enclosure(s) (please Identify below):
  1. Replacement Attachment to Request for Inter-Partes Re-Examination of U.S. Patent No. 6,857,001;
  2. Certificate of Mailing By Express Mail No.: EV 978 428 202 US;
  3. Return Post Card

Remarks

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm Name	DLA Piper US LLP		
Signature	Ronald L. Yin		
Printed name	Ronald L. Yin		
Date	February 8, 2008	Reg. No.	27,607

CERTIFICATE OF TRANSMISSION/MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage via "EXPRESS MAIL No. EV 978 428 202 US in an envelope addressed to: Central Reexamination Unit (CRU), Box Inter Partes Reexam, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on the date shown below:

Signature	Ronald L. Yin		
Typed or printed name	Ronald L. Yin	Date	February 8, 2008

This collection of information is required by 37 CFR 1.5. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to 2 hours to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: **Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Hitz et al.	
U.S. Patent No.	6,857,001	Issued: February 15, 2005
Filed:	June 7, 2002	Control Number: 95/000,324
Docket No.:	347155-29	

Title: MULTIPLE CONCURRENT ACTIVE FILE SYSTEM

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage Via Express Mail No. EV 978 428 202 US in an envelope addressed to: Central Reexamination Unit (CRU), ATTN: "Box Inter Partes Reexam", Commissioner for Patents, United States Patent and Trademark Office, P.O. Box 1450, Alexandria, VA. 22313-1450 on:
February 8, 2008
Ronald L. Yin

* * *

RESPONSE TO NOTICE OF FAILURE TO COMPLY WITH INTER PARTES REEXAMINATION REQUEST FILING REQUIREMENTS (37 CFR 1.915(D))

Central Reexamination Unit (CRU)
ATTN: "Box Inter Partes Reexam"
Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA. 22313-1450

Sir:

In response to the notice mailed January 11, 2008 ("Notice"), Requester has attached herewith a Replacement Attachment for Re-Examination complying with 37 CFR 1.915. Specifically, in the Notice, the PTO stated that the request failed to comply with the requirements of 37 CFR 1.915(b)(3) because the request did not include a detailed explanation of how each of the eight references listed in the Notice applied to claims 1-63 of U.S. patent 6,857,001 for which reexamination was requested. This has also been rectified by the accompanying Replacement Attachment. Requester notes that the Notice alleged that the Sun reference did not include a detailed explanation of how that reference applied to claims 1-63.

However, the Sun reference was specifically identified and explained in detail as to how it can be applied to claims 1-59. See pages 94-107 of the Original Attachment.

A copy of this response as well as the Replacement Attachment is served on the patent owner as provided in 37 CFR 1.33(c). The name and address of the party served and the date of service are:

Swernofsky Law Group PC
P.O. Box 390013
Mountain View, CA 94039-0013

Served On: February 8, 2008

Any fee due for this reexamination may be charged to Deposit Account No. 07-1896.

Respectfully submitted,

DLA PIPER US LLP

Date: February 8, 2008

By: Ronald L. Yin
Ronald L. Yin
Reg. No. 27,607

Attorneys for Requester

Ronald L. Yin
DLA Piper US LLP
2000 University Avenue
East Palo Alto, CA 94303-2248
650-833-2437 (Direct)
650-833-2000 (Main)
650-833-2001 (Facsimile)
ronald.yin@dlapiper.com

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant:	Hitz et al.	
U.S. Patent No.	6,857,001	Issued: February 15, 2005
Filed:	June 7, 2002	Control Number: 95/000,324
Docket No.:	347155-29	

Title: MULTIPLE CONCURRENT ACTIVE FILE SYSTEM

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage Via Express Mail No. EV 978 428 202 US in an envelope addressed to: Commissioner of Patents, MS Inter Partes REEXAM, P.O. Box 1450, Alexandria, VA 22313-1450, on:
February 8, 2008

Ronald L. Yin

* * *

REPLACEMENT ATTACHMENT TO REQUEST FOR INTER-PARTES RE-EXAMINATION OF U.S. PATENT NO. 6,857,001

Mail Stop *Inter Partes* Reexam
Commissioner for Patents
P.O. Box 1450
Alexandria, VA. 22313-1450

Sir:

Pursuant to 35 U.S.C. §§ 311-318 and 37 CFR § 1.903-1.997, this is a request for inter-partes reexamination of United States Patent No. 6,857,001 which issued on February 15, 2005 to Hitz et al. (the "'001 Patent").

I. CLAIMS FOR WHICH REEXAMINATION IS REQUESTED

Reexamination is requested of Claims 1-63 of the '001 Patent in view of the prior art listed on the Citation of Prior Art under 37 CFR § 1.501 and 35 U.S.C. § 301 which is submitted with the Request for Reexamination.

II. EXPLANATION OF PERTINENCE AND MANNER OF APPLYING CITED PRIOR ART TO EVERY CLAIM FOR WHICH REEXAMINATION IS REQUESTED

Introduction

One or more prior art references submitted with the Citation of Prior Art renders claims 1-63 of the '001 Patent either anticipated under 35 USC 102(a), (b) or (e) or unpatentable under 35 USC 103 so that substantial new questions of patentability of the claims of the '001 Patent have been raised by this request for reexamination. For each claim for which reexamination is sought, a specific citation of the prior art or the combination of the prior art pertinent to the claim and a description of the relevancy of that prior art to the claim are set forth blow in greater detail. None of the prior art cited in the Citation of Prior Art was submitted to the examiner or considered by the examiner during the prosecution of the '001 Patent.

Statements Identifying Substantial New Questions of Patentability

The following Substantial New Question (SNQ) of Patentability are raised with respect to the claims of the '001 Patent in view of the prior art listed on the Citation of Prior Art.

SNQ 1. A substantial new question of patentability as to claims 1-63 is raised by the references Hitz and Ylonen.

SNQ 2. A substantial new question of patentability as to claims 1-63 is raised by the reference VxFS.

SNQ 3. A substantial new question of patentability as to claims 1-63 is raised by the reference Siddha.

SNQ 4. A substantial new question of patentability as to claims 1-59 is raised by the reference Sun.

SNQ 5. A substantial new question of patentability as to claims 1-63 is raised by the reference Siddha Report.

SNQ 6. A substantial new question of patentability as to claims 1-6, 10-12, 14-15, 20-25, 29-33, 39-44, and 48-52 is raised by the reference Czezatke.

SNQ 7. A substantial new question of patentability as to claims 1-5, 10-12, 20-24, 29-31, 39-43 and 48-50 is raised by the reference LSI Logic Whitepaper.

SNQ 8. A substantial new question of patentability as to claims 1-5, 10-12, 20-24, 29-31, 39-43, 48-50 is raised by the reference Osorio.

SNQ 9. A substantial new question of patentability as to claims 1, 10, 20, 29, 39 and 48 is raised by the reference Grummon.

SNQ 10. A substantial new question of patentability as to claims 1, 10, 20, 29, 39 and 48 is raised by the reference Allen.

SNQ 11. A substantial new question of patentability as to claims 1, 10, 20, 29, 39 and 48 is raised by the reference Lim.

Explanation of How Each SNQ Is Raised

1. Ylonen section 3 teaches maintaining snapshots that are a transaction-consistent copy of a database through shadow paging techniques. Fig. 2 in Ylonen section 3.1 further illustrates maintaining multiple snapshots. Ylonen section 3.4 teaches that such snapshots are a copy-on-write copy of the database. "From the user's point of view, each snapshot is an independent copy of the database. It is possible to make modifications to the copy, and these modifications will not affect other copies." (page 13) This is further illustrated in Ylonen Fig.3. Thus, the writable snapshots of Ylonen initially access shared data and grow apart over time. For example, Current State 3 and Current State 2 in Fig. 3 above initially access data of parent Snapshot 25. Ylonen section 3.4 teaches that "copies diverge as more modifications are made." Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz. Thus, this combination of Ylonen and Hitz raises an SNQ with respect to at least claim 1 of the '001 Patent. A detailed explanation of the application of these references to this SNQ with respect to the other claims is set forth herein below under the First Basis of Invalidity.

2. VxFS discloses a file system. Chapter 8 discloses storage checkpoints, a copy-on-write snapshot of the file system. VxFS storage checkpoints can be mounted as writable checkpoints

using the –o rw or –o remount options. (See p. 86, p. 89.) When such a storage checkpoint is mounted as writable, it initially shares data with the original file system through copy-on-write techniques. For instance, at pp. 84-85, VxFS teaches that "[t]he Storage Checkpoint presents the exact image of the file system by finding the data from the primary fileset. . . . This is called the copy-on-write technique, which allows the Storage Checkpoint to preserve the image of the primary fileset when the Storage Checkpoint is taken." Once a storage checkpoint is mounted as writable, it is inherent that it becomes an independent active file system. For instance, VxFS at p. 86 states that "[y]ou can mount, access, and write to a data Storage Checkpoint just as you would to a file system." Furthermore, VxFS at p. 89 warns that such a writable checkpoint will diverge from a read-only storage checkpoint when it is written to. Changes made to a writable storage checkpoint are not reflected in the original file system. Thus, VxFS raises an SNQ with respect to at least claim 1 of the '001 Patent. A detailed explanation of the application of this reference to this SNQ with respect to the other claims is set forth herein below under the Second Basis of Invalidity.

3. Siddha generally and section 3.1.3 specifically teaches a file system with a multiple snapshot capability, wherein the snapshots are established using copy-on-write techniques so as not to overwrite data. Siddha section 3.1.3 further teaches that snapshots can be made writable, establishing a diverging snapshot tree of multiple active file systems, as shown in Fig. 4. Snapshot Ck' is even labeled as "Writable Snapshot #1." Siddha teaches that, because of the copy-on-write nature of the snapshots, active file systems formed by mounting snapshots as writable initially share data sets with parent file systems and diverge over time. Thus, in Fig. 4, writable snapshot Ck' initially shares data with snapshot Ck, which represents the state of the original file system at time k. Snapshot Ck'-1, representing the state of the new file system at a later time, does not share changed data with snapshot Ck-1, which represents the state of the original file system. Thus, Siddha raises an SNQ with respect to at least claim 1 of the '001 Patent. A detailed explanation of the application of this reference to this SNQ with respect to the other claims is set forth herein below under the Third Basis of Invalidity.

4. Sun teaches at pp. 1-5 through 1-6 a master volume (volume containing the original data) and a shadow volume (volume that is the copy of the master volume). The shadow volume can be independent (produced via a full volume copy operation) or dependent. The dependent shadow volume is a point-in-time copy that relies on the master for all unmodified data blocks.

When a data block in the master volume is modified, a copy is placed in the dependent shadow, so as to preserve point-in-time consistency of the shadow block. Thus, dependent shadow volumes use a copy-on-write technique. Sun further teaches at p. 1-6 that "[o]nce the shadow volume is created, you can read from and write to it. After you write to it, the shadow volume data is unique and does not necessarily match the master volume. A bitmap volume file tracks the differences between the two." Thus, Sun raises an SNQ with respect to at least claim 1 of the '001 Patent. A detailed explanation of the application of this reference to this SNQ with respect to the other claims is set forth herein below under the Fourth Basis of Invalidity.

5. Siddha Report generally and section 1.1 specifically teaches a file system with a multiple snapshot capability, wherein the snapshots are established using copy-on-write techniques so as not to overwrite data, as shown in Fig. 1. Siddha Report section 2.1.3 further teaches that snapshots can be made writable, establishing a diverging snapshot tree of multiple active file systems, as depicted in Fig. 2.3. Snapshot Ck' is even labeled as "Writable Snapshot #1." Siddha Report teaches that, because of the copy-on-write nature of the snapshots, active file systems formed by mounting snapshots as writable initially share data sets with parent file systems and diverge over time. Thus, in Fig. 2.3, writable snapshot Ck' initially shares data with snapshot Ck, which represents the state of the original file system at time k. Snapshot Ck'-1, representing the state of the new file system at a later time, does not share changed data with snapshot Ck-1, which represents the state of the original file system. Thus, Siddha Report raises an SNQ with respect to at least claim 1 of the '001 Patent. A detailed explanation of the application of this reference to this SNQ with respect to the other claims is set forth herein below under the Fifth Basis of Invalidity.

6. Czezatke teaches a Log-Structured File System, which is a method for operating data storage. Czezatke further teaches maintaining plural active file systems through the mechanism of cloning or writable snapshots. See 1. Introduction – page 1. The Czezatke file system is implemented using copy-on-write techniques, where new data never overwrites old data. When the Czezatke file system makes a snapshot or clone, the snapshot or clone inherently initially shares its data with the parent file system. A snapshot directly shares data with its parent active file system. And a clone shares data with its parent snapshot. Thus, when a clone (which is an active file system) is initially created, it will share data with the grandparent active file system to the same extent that its parent snapshot shares data with the grandparent active file system.

When the data in an active file system changes, those changes are not reflected in the clone, which is based on a fixed-image snapshot. And, when the data in a clone changes, those changes are not reflected in the parent snapshot or in the active file system from which that snapshot was generated. See Sec. 3.3 (discussing tracking which data blocks are shared with other clones). This feature of writable snapshots makes it possible to explore the "what-if scenarios," as described by Czezatke in the paragraph reproduced above. The '001 patent specification describes an advantage to writable snapshots that is substantively identical to the description in Czezatke. "It would become possible to make changes to an 'experimental' version of the file system." Col. 1:48-50. Thus, Czezatke raises an SNQ with respect to at least claim 1 of the '001 Patent. A detailed explanation of the application of this reference to this SNQ with respect to the other claims is set forth herein below under the Sixth Basis of Invalidity.

7. LSI Logic Whitepaper discloses on page 4, a method of making writable snapshots of logical volumes, wherein such a volume is representative of a file system. When the LSI Logic SANtricity snapshot feature makes a writable snapshot, the snapshot inherently initially shares its data with the parent file system. The writable snapshot directly shares and data with its parent active file system, referred to as the "base volume" Thus, when a writable snapshot (which is an active file system) is initially created, it will share data with its base volume (which remains an active file system). As data is written to the writable snapshot, it will diverge from the base volume, with changes in one not reflected in the other. Thus, LSI Logic Whitepaper raises an SNQ with respect to at least claim 1 of the '001 Patent. A detailed explanation of the application of this reference to this SNQ with respect to the other claims is set forth herein below under the Seventh Basis of Invalidity.

8. Osorio discloses a method for operating data storage including maintaining file systems using copy-on-write snapshots that can be written to. Pages 3-4 disclose copy-on-write snapshots as a "copy image of storage devices or file systems." Page 7 further teaches that the snapshot can be converted into another active file system that initially shares data with the original system, but diverges over time, with changes made to each of the active file systems not reflected in the other active file system: "When a second host can read and write the snapshot image, the second host can start a second Oracle instance. . . . For this usage, the snapshot area essentially is a new database started with a copy of the original database." Thus, Osorio raises an SNQ with respect to at least claim 1 of the '001 Patent. A detailed explanation of the

application of this reference to this SNQ with respect to the other claims is set forth herein below under the Eighth Basis of Invalidity.

9. Grummon discloses a file system using copy-on-write snapshots and having writable snapshots. Fig. 3 discloses a file system 300. Read-Write On-Line Container 310 represents the active file system while Read-Write Snapshot Container 308 represents the writable snapshot. Col. 7: 11-16. The snapshot and active file system initially share data. See e.g. Col. 6: 37-47. As data is written to container 308, changes are not shared. See e.g. Col. 7:17-63. Thus, Grummon raises an SNQ with respect to at least claim 1 of the '001 Patent. A detailed explanation of the application of this reference to this SNQ with respect to the other claims is set forth herein below under the Ninth Basis of Invalidity.

10. Allen teaches a file system for viewing selected versions of the files stored in a versioned object base (VOB). See column 6, lines 16-18, and Fig. 2. A virtual file system is provided that allows users to have a private view of the files. Col. 6:49-58. Allen further teaches "branching" of the file trees, with each of the branches undergoing independent evolution from parent and sister branches. Col. 6:59-7:22, Fig. 3. As shown in Fig. 3, a branch initially shares data with the parent, but diverges over time, with changes not shared with the other branches. Therefore, Allen teaches plural active file trees, wherein changes made to each file tree are not reflected in the file trees with which the first tree initially shares data. Thus, Allen raises an SNQ with respect to at least claim 1 of the '001 Patent. A detailed explanation of the application of this reference to this SNQ with respect to the other claims is set forth herein below under the Tenth Basis of Invalidity.

11. Lim teaches a computer system state checkpoint mechanism. Multiple checkpoints can be taken. *See e.g.* Col. 6:45-65, 8:5-7. The checkpoints can be mounted on another virtual machine and used as an initial state, diverging over time as the virtual machine operates. *See e.g.* Col. 7: 6-18, 8:8-14. The checkpoints inherently include files. Therefore, Lim teaches plural active file systems, wherein changes made to each file tree are not reflected in the file trees with which the first tree initially shares data. Thus, Lim raises an SNQ with respect to at least claim 1 of the '001 Patent. A detailed explanation of the application of this reference to this SNQ with respect to the other claims is set forth herein below under the Eleventh Basis of Invalidity.

Invalidity

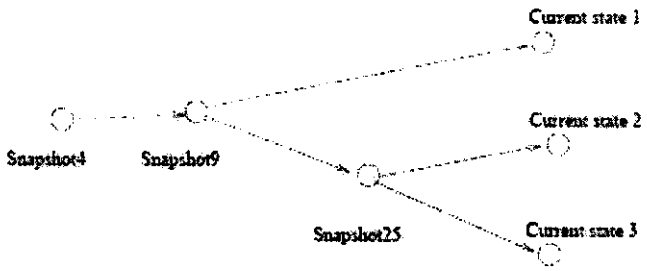
First Basis of Invalidity

The references applicable to the first basis of invalidity are:

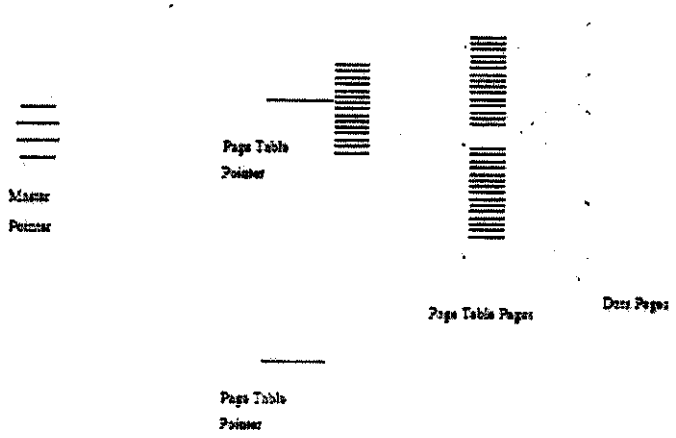
1. Hitz et. al, *File System Design For An NFS File Server Appliance*, TR3002, USENIX January 19, 1994 (Hereinafter: "Hitz")

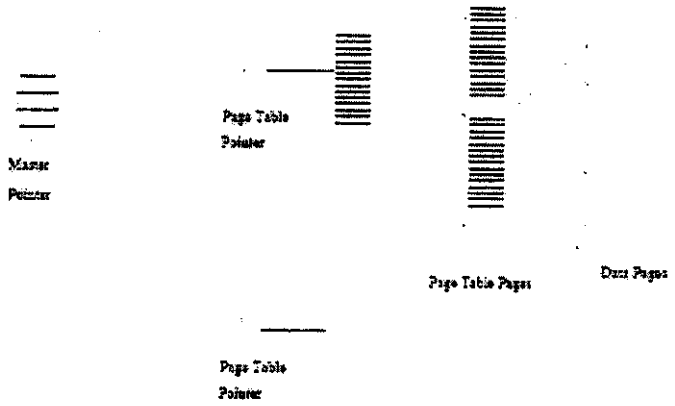
2. Ylonen et. al, *Concurrent Shadow Paging: Snapshots, Read-Only Transactions, and On-The-Fly Multilevel Incremental Dumping*, TKO-B104, Laboratory of Information Processing Science at Helsinki University of Technology, 1993. (Hereinafter: "Ylonen"). Although this document is undated on its face, it is referenced in Ylonen et. al, *Concurrent Shadow Paging: Fine-Granularity Locking with Support for Extended Lock Modes and Early Releasing of Locks*, Laboratory of Information Processing Science at Helsinki University of Technology (see Footnote [17], page 28), referencing Ylonen as being published in 1993.

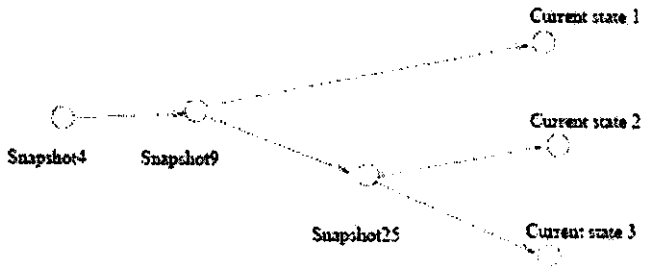
The pertinence and manner of applying Ylonen and Hitz to claims 1-63 for which re-examination is requested is as follows:

Claims of '001 Patent	Ylonen and Hitz
1. A method of operating data storage, the method including maintenance of plural active file systems, wherein each of the active file systems initially access data shared with another of the active file systems, and wherein changes made to each of the active file systems are not reflected in the active file system with which the changed active file system shares the data.	Ylonen section 3 teaches maintaining snapshots that are a transaction-consistent copy of a database through shadow paging techniques. Fig. 2 in Ylonen section 3.1 further illustrates maintaining multiple snapshots. Ylonen section 3.4 teaches that such snapshots are a copy-on-write copy of the database. "From the user's point of view, each snapshot is an independent copy of the database. It is possible to make modifications to the copy, and these modifications will not affect other copies." (page 13) This is further illustrated in Ylonen Fig.3:  Thus, the writable snapshots of Ylonen initially access shared data and grow apart over time. For example, Current State 3 and Current State 2 in Fig. 3 above initially access data of

	parent Snapshot 25. Ylonen section 3.4 teaches that "copies diverge as more modifications are made." Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
2. A method as in claim 1, wherein when a second active file system is created based on a first active file system, the first active file system and the second active file system initially share data.	Ylonen teaches in section 3.4 and Fig. 3 that a second active database version (e.g. Current State 2 in Fig. 3) is created based on a first active database version (Current State 1), with the two versions initially sharing data (Snapshot 9). Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
3. A method as in claim 2, wherein when changes are made to the first active file system, modified data is recorded in the first active file system in a location that is not shared with the second active file system.	Ylonen section 3.4 teaches that "From the user's point of view, each snapshot is an independent copy of the database. It is possible to make modifications to the copy, and these modifications will not affect other copies." (page 12). Ylonen Fig. 5 and the text in section 3.5 further teach that active database versions do not share changed data locations.

	 Figure 5: Shadow paging file structure with a master pointer and multiple versions. In Fig. 5 each Page Table Pointer represents an active version of the database. As taught in Ylonen section 2, the shadow paging technique (also known as copy-on-write) requires that changes in Data Pages are recorded in Page Table Pages. As database versions disclosed in Ylonen Fig. 5 diverge over time the respective Page Table Pages are not shared. In addition, changed Data Pages are recorded in different locations using shadow paging / copy-on-write techniques. Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
4. A method as in claim 2, wherein when changes are made to the second active file system, modified data is recorded in the second active file system in a location that is not shared with the first active file system.	Ylonen section 3.4 teaches that "From the user's point of view, each snapshot is an independent copy of the database. It is possible to make modifications to the copy, and these modifications will not affect other copies." (page 12). Ylonen Fig. 5 and the text in section 3.5 further teach that active database versions do not share changed data locations.

	 Figure 5: Shadow paging file structure with a master pointer and multiple versions. In Fig. 5 each Page Table Pointer represents an active version of the database. As taught in Ylonen section 2, the shadow paging technique (also known as copy-on-write) requires that changes in Data Pages are recorded in Page Table Pages. As database versions disclosed in Ylonen Fig. 5 diverge over time the respective Page Table Pages are not shared. In addition, changed Data Pages are recorded in different locations using shadow paging / copy-on-write techniques. Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
5. A method as in claim 1, wherein snapshots are made of ones of the plural active file systems, each snapshot forming an image of its respective active file system at a past consistency point.	Ylonen section 3 teaches maintaining snapshots that are a transaction-consistent copy of a database through shadow paging techniques. Fig. 2 in Ylonen section 3.1 further illustrates maintaining multiple snapshots. Ylonen section 3.4 teaches modifying such snapshots that are a copy-on-write copy of the database. "From the user's point of view, each snapshot is an independent copy of the database. It is possible to make modifications to the copy,

	and these modifications will not affect other copies." (page 12). This is further illustrated in Ylonen Fig.3:  Fig. 3 teaches that Snapshot 9 is an image of active database version Current State 1 at a past consistency point, while Snapshot 25 is an image of active database version Current State 2. Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
6. A method as in claim 5, wherein each snapshot includes a complete hierarchy for file system data, separate and apart from active file system data for the plural active file systems.	Ylonen Fig. 5 and the text in section 3.5 teach that each permanent snapshot includes a hierarchy of system data, such as Page Table Pointer and Page Table Pages. Some of these permanent snapshots are active database versions.

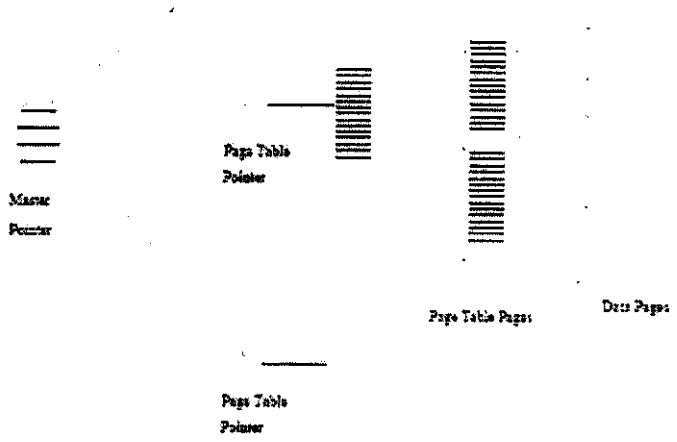


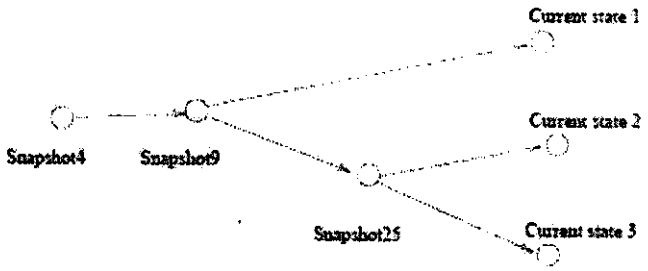
Figure 5: Shadow paging file structure with a master pointer and multiple versions.

Furthermore, Hitz Figs. 3 and 4 at p. 11 teach that an active file system and snapshots have separate file system data hierarchies.

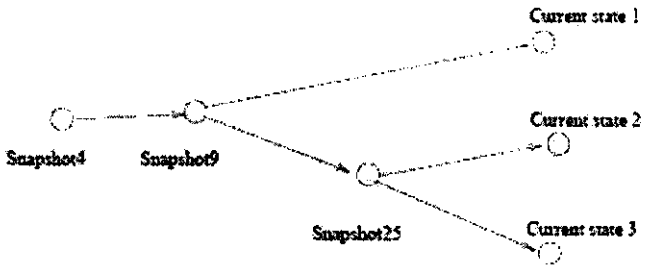
Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor.

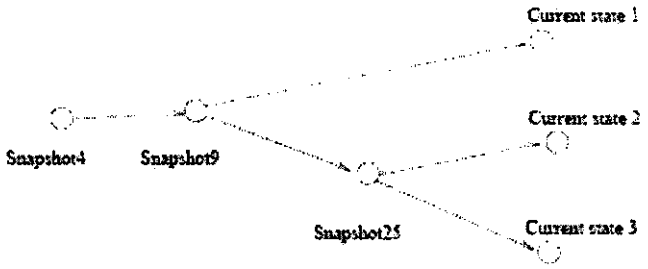
Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.

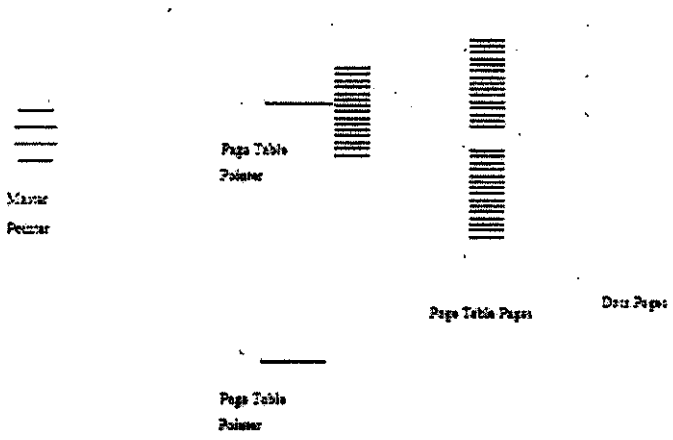
7. A method as in claim 5, wherein at least one of the snapshots is converted into a new active file system.	Ylonen Fig. 3 teaches snapshots converted to active database versions.

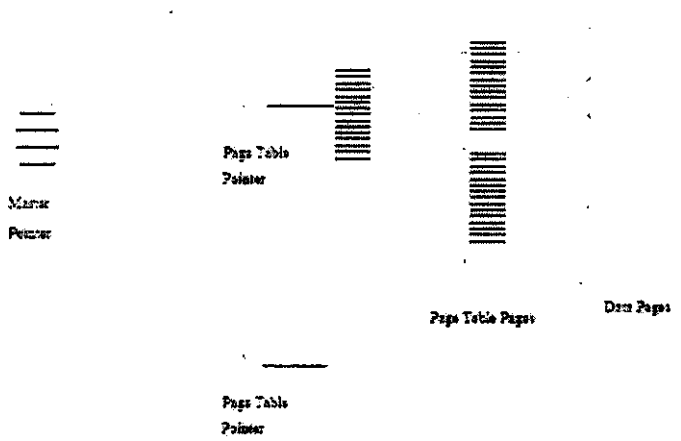


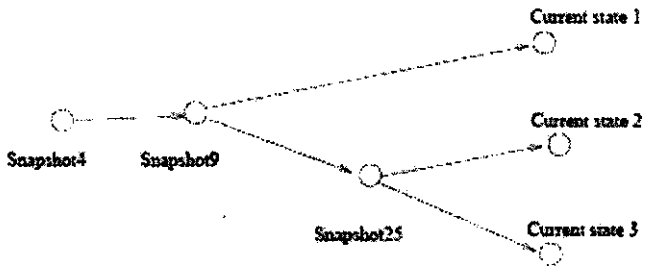
Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-

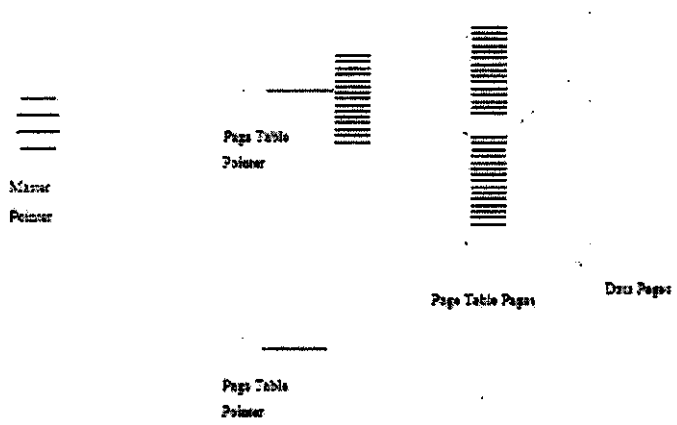
	write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
8. A method as in claim 7, wherein the one of the snapshots is converted by making the one of the snapshots writable.	Ylonen section 3.4 teaches allowing modifications to certain snapshots so as to covert them to writable. Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
9. A method as in claim 8, wherein snapshot pointers from any of the active file systems to the new active file system are severed.	Ylonen section 3.4 and Fig. 3 teach that active database versions are independent and unsynchronized, diverging over time. Fig. 3 further shows that there are no snapshot pointers between any of the active database versions. Snapshot pointers point only from snapshots to their respective active database versions.  Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file

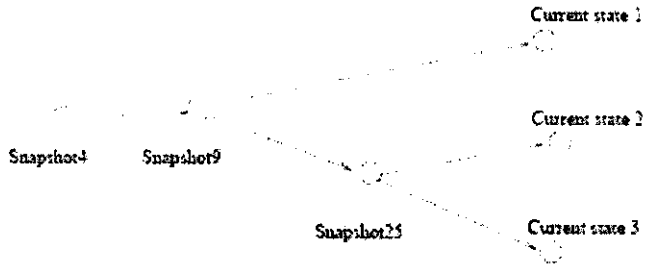
	system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
10. A method of creating plural active file systems, comprising the steps of: making a snapshot of a first active file system, the snapshot initially sharing data with the first active file system; and converting the snapshot to a second active file system by making the snapshot writable, with changes made to the first active file system not reflected in the second active file system, and with changes made to the second active file system not reflected in the first active file system.	Ylonen section 3 teaches maintaining snapshots that are a transaction-consistent copy of a database through shadow paging techniques. Fig. 2 in Ylonen section 3.1 further illustrates maintaining multiple snapshots. Ylonen section 3.4 teaches that such snapshots are a copy-on-write copy of the database. "From the user's point of view, each snapshot is an independent copy of the database. It is possible to make modifications to the copy, and these modifications will not affect other copies." (page 12). This is further illustrated in Ylonen Fig.3:  Thus, the writable snapshots of Ylonen initially access shared data and grow apart over time. For example, Current State 3 and Current State 2 in Fig. 3 above initially access data of parent Snapshot 25. Ylonen section 3.4 teaches that "copies diverge as more modifications are made." Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily

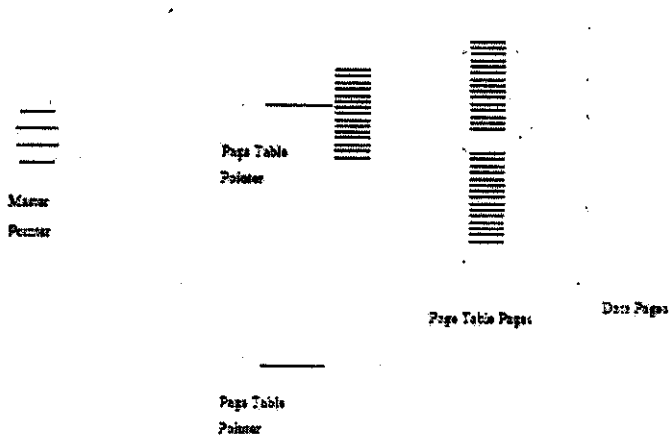
	appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
11. A method as in claim 10, wherein when changes are made to the first active file system, modified data is recorded in the first active file system in a location that is not shared with the second active file system.	Ylonen section 3.4 teaches that "From the user's point of view, each snapshot is an independent copy of the database. It is possible to make modifications to the copy, and these modifications will not affect other copies." (page 12). Ylonen Fig. 5 and the text in section 3.5 further teach that active database versions do not share changed data locations.  Figure 5: Shadow paging file structure with a master pointer and multiple versions. In Fig. 5 each Page Table Pointer represents permanent snapshots, some or all of which are active versions of the database. As taught in Ylonen section 2, the shadow paging technique (also known as copy-on-write) requires that changes in Data Pages are recorded in Page Table Pages. As database versions disclosed in Ylonen Fig. 5 diverge over time the respective Page Table Pages are not shared. In addition, changed Data Pages are recorded in different locations using shadow paging / copy-on-write techniques. Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily

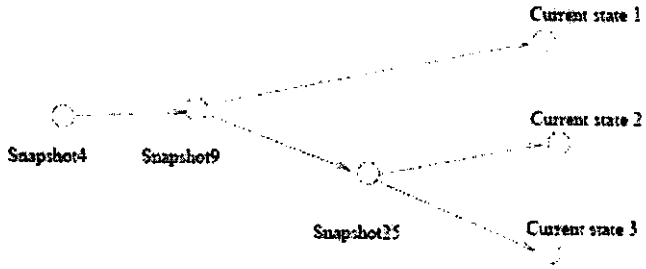
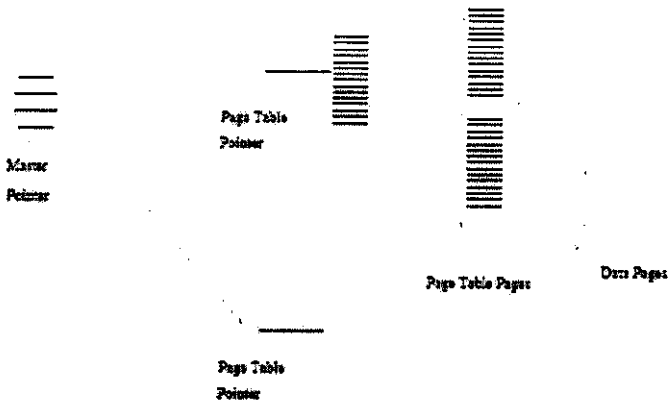
	appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
12. A method as in claim 10, wherein when changes are made to the second active file system, modified data is recorded in the second active file system in a location that is not shared with the first active file system.	Ylonen section 3.4 teaches that "From the user's point of view, each snapshot is an independent copy of the database. It is possible to make modifications to the copy, and these modifications will not affect other copies." (page 12) Ylonen Fig. 5 and the text in section 3.5 further teach that active database versions do not share changed data locations.  Figure 5: Shadow paging file structure with a master pointer and multiple versions. In Fig. 5 each Page Table Pointer represents permanent snapshots, some or all of which are active versions of the database. As taught in Ylonen section 2, the shadow paging technique (also known as copy-on-write) requires that changes in Data Pages are recorded in Page Table Pages. As database versions disclosed in Ylonen Fig. 5 diverge over time the respective Page Table Pages are not shared. In addition, changed Data Pages are recorded in different locations using shadow paging / copy-on-write techniques. Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily

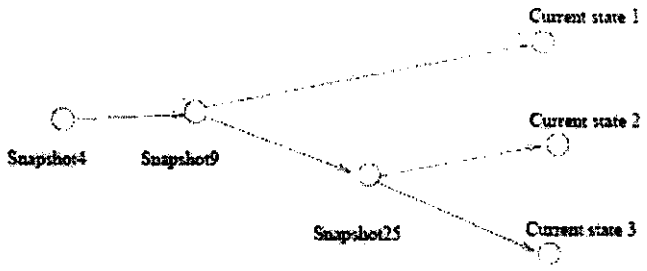
	appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
13. A method as in claim 10, further comprising the step of severing any snapshot pointers from the first active file system to the second active file system.	Ylonen section 3.4 and Fig. 3 teach that active database versions are independent and unsynchronized, diverging over time. Fig. 3 further shows that there are no snapshot pointers between any of the active database versions. Snapshot pointers point only from snapshots to their respective active database versions.  Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
14. A method as in claim 10, further comprising the steps of making snapshots of ones of the plural active file systems.	Fig. 3 teaches that Snapshot 9 is a snapshot of active database version Current State 1 while Snapshot 25 is a snapshot of active database version Current State 2.  Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file

	system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
15. A method as in claim 14, wherein each snapshot includes a complete hierarchy for file system data, separate and apart from active file system data for the plural active file systems.	Ylonen Fig. 5 and the text in section 3.5 teach that each permanent snapshot includes a hierarchy of system data, such as Page Table Pointer and Page Table Pages. Some of these permanent snapshots are active database versions.  Figure 5: Shadow paging file structure with a master pointer and multiple versions. Furthermore, Hitz Figs. 3 and 4 at p. 11 teach that an active file system and snapshots have separate file system data hierarchies. Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily

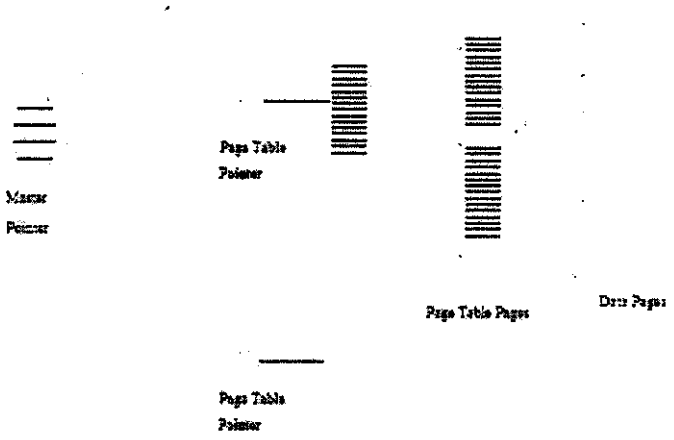
	appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
16. A method as in claim 10, further comprising the steps of: making a new snapshot of the first active file system, the new snapshot initially sharing data with the first active file system; converting the new snapshot to a third active file system by making the new snapshot writable, with changes made to the first active file system or the second active file system not reflected in the third active file system.	Ylonen Fig. 3 teaches making Snapshot 25 that initially shares data with active database version Current State 1 as both share the same parent Snapshot 9. Snapshot 25 is converted to active database version Current State 2. Changes made in each of Current State 1, Current State 2 and Current State 3 are not reflected in the other active database versions.  Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
17. A method as in claim 16, wherein when changes are made to the first active file system or the second active file system, modified data is recorded in a location that is not shared with the third active file system.	Ylonen section 3.4 teaches that "From the user's point of view, each snapshot is an independent copy of the database. It is possible to make modifications to the copy, and these modifications will not affect other copies." (page 12). Ylonen Fig. 5 and the text in section 3.5 further teach that active database versions do not share changed data locations.

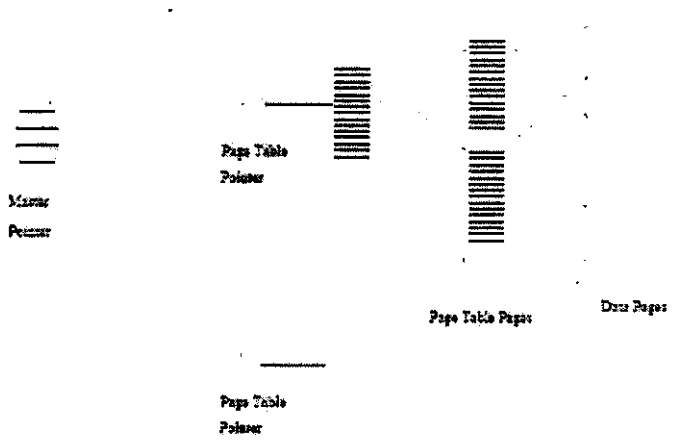
	 Figure 5: Shadow paging file structure with a master pointer and multiple versions. In Fig. 5 each Page Table Pointer represents permanent snapshots, some or all of which are active versions of the database. As taught in Ylonen section 2, the shadow paging technique (also known as copy-on-write) requires that changes in Data Pages are recorded in Page Table Pages. As database versions disclosed in Ylonen Fig. 5 diverge over time the respective Page Table Pages are not shared. In addition, changed Data Pages are recorded in different locations using shadow paging / copy-on-write techniques. Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases." and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
18. A method as in claim 10, further comprising the steps of: making a new snapshot of the second active file system, the new snapshot initially sharing data with the second active file system; converting the new	Ylonen Fig. 3 teaches making Snapshot 25 that initially shares data with active database version Current State 1 as both share the same parent Snapshot 9. Snapshot 25 is converted to active database version Current State 2. Changes made in each of Current State 1, Current State 2 and Current State 3 are not reflected in the other active database versions.

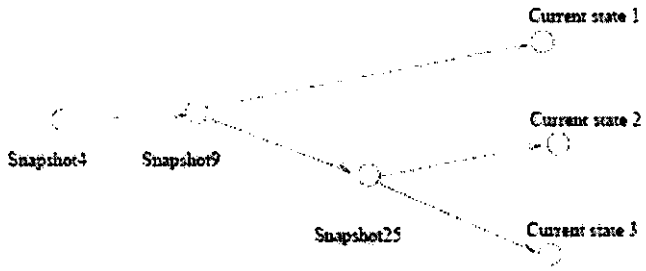
snapshot to a third active file system by making the new snapshot writable, with changes made to the first active file system or the second active file system not reflected in the third active file system.	 Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
19. A method as in claim 18, wherein when changes are made to the first active file system or the second active file system, modified data is recorded in a location that is not shared with the third active file system.	Ylonen section 3.4 teaches that "From the user's point of view, each snapshot is an independent copy of the database. It is possible to make modifications to the copy, and these modifications will not affect other copies." (page 12). Ylonen Fig. 5 and the text in section 3.5 further teach that active database versions do not share changed data locations.  **Figure 5: Shadow paging file structure with a master pointer and multiple versions.** In Fig. 5 each Page Table Pointer represents permanent

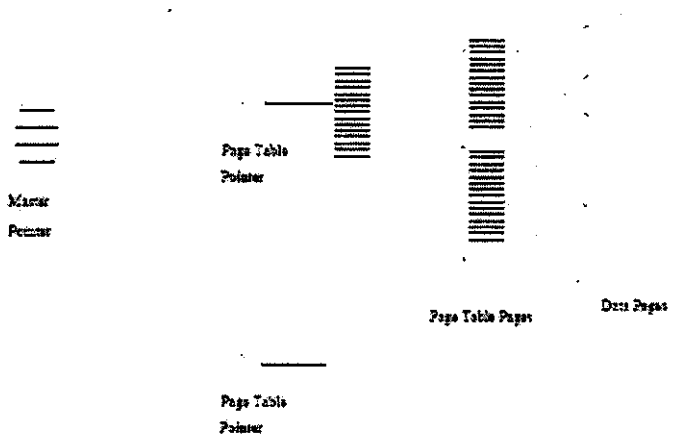
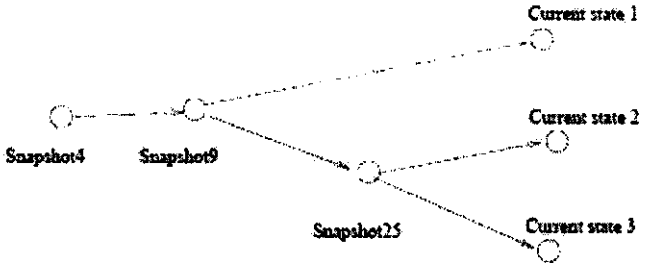
	snapshots, some or all of which are active versions of the database. As taught in Ylonen section 2, the shadow paging technique (also known as copy-on-write) requires that changes in Data Pages are recorded in Page Table Pages. As database versions disclosed in Ylonen Fig. 5 diverge over time the respective Page Table Pages are not shared. In addition, changed Data Pages are recorded in different locations using shadow paging / copy-on-write techniques. Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
20. A memory storing information including instructions, the instructions executable by a processor to operate data storage, the instructions comprising steps to maintain plural active file systems, wherein each of the active file systems initially access data shared with another of the active file systems, and wherein changes made to each of the active file systems are not reflected in the active file system with which the changed active file system shares the data.	Ylonen section 3 teaches maintaining snapshots that are a transaction-consistent copy of a database through shadow paging techniques. Fig. 2 in Ylonen section 3.1 further illustrates maintaining multiple snapshots. Ylonen section 3.4 teaches that such snapshots are a copy-on-write copy of the database. "From the user's point of view, each snapshot is an independent copy of the database. It is possible to make modifications to the copy, and these modifications will not affect other copies." (page 12). This is further illustrated in Ylonen Fig.3:  Thus, the writable snapshots of Ylonen initially access shared data and grow apart over time. For example, Current State 3 and Current State 2 in Fig. 3 above initially access data of parent Snapshot 25. Ylonen section 3.4 teaches that "copies diverge as more modifications are made." Ylonen further

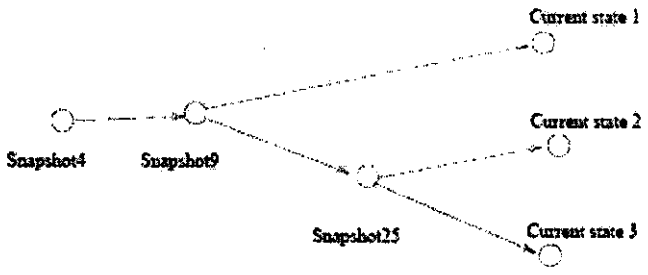
	teaches in section 2, (page 5) that the pages of data are stored on disks operable under the control of a CPU, which inherently operates the data storage device by executing instructions. Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
21. A memory as in claim 20, wherein when a second active file system is created based on a first active file system, the first active file system and the second active file system initially share data.	Ylonen teaches in section 3.4 and Fig. 3 that a second active database version (e.g. Current State 2 in Fig. 3) is created based on a first active database version (Current State 1), with the two versions initially sharing data (Snapshot 9). Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
22. A memory as in claim 21, wherein when changes are made to the first active file system, modified data is recorded in the first active file system in a location that is not shared with the second active file system.	Ylonen section 3.4 teaches that "From the user's point of view, each snapshot is an independent copy of the database. It is possible to make modifications to the copy, and these modifications will not affect other copies." (page 12). Ylonen Fig. 5 and the text in section 3.5 further teach that active database versions do not share changed data locations.

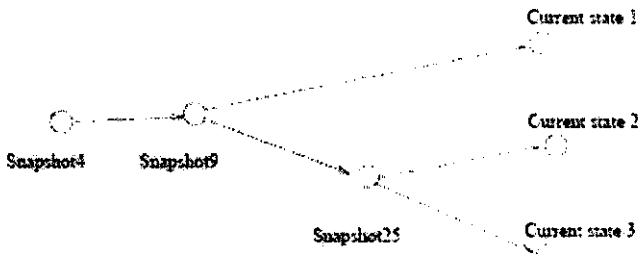
	 Figure 5: Shadow paging file structure with a master pointer and multiple versions. In Fig. 5 each Page Table Pointer represents an active version of the database. As taught in Ylonen section 2, the shadow paging technique (also known as copy-on-write) requires that changes in Data Pages are recorded in Page Table Pages. As database versions disclosed in Ylonen Fig. 5 diverge over time the respective Page Table Pages are not shared. In addition, changed Data Pages are recorded in different locations using shadow paging / copy-on-write techniques. Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
23. A memory as in claim 21, wherein when changes are made to the second active file system, modified data is recorded in the second active file system in a location that is not shared with the first active file system.	Ylonen section 3.4 teaches that "From the user's point of view, each snapshot is an independent copy of the database. It is possible to make modifications to the copy, and these modifications will not affect other copies." (page 12). Ylonen Fig. 5 and the text in section 3.5 further teach that active database versions do not share changed data locations.

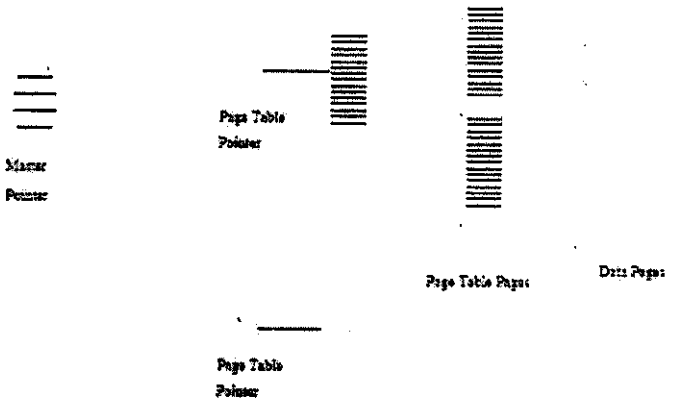
	 Figure 5: Shadow paging file structure with a master pointer and multiple versions. In Fig. 5 each Page Table Pointer represents an active version of the database. As taught in Ylonen section 2, the shadow paging technique (also known as copy-on-write) requires that changes in Data Pages are recorded in Page Table Pages. As database versions disclosed in Ylonen Fig. 5 diverge over time the respective Page Table Pages are not shared. In addition, changed Data Pages are recorded in different locations using shadow paging / copy-on-write techniques. Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
24. A memory as in claim 20, wherein snapshots are made of ones of the plural active file systems, each snapshot forming an image of its respective active file system at a past consistency point.	Ylonen section 3 teaches maintaining snapshots that are a transaction-consistent copy of a database through shadow paging techniques. Fig. 2 in Ylonen section 3.1 further illustrates maintaining multiple snapshots. Ylonen section 3.4 teaches modifying such snapshots that are a copy-on-write copy of the database. "From the user's point of view, each snapshot is an independent copy of the database. It is possible to make modifications to the copy,

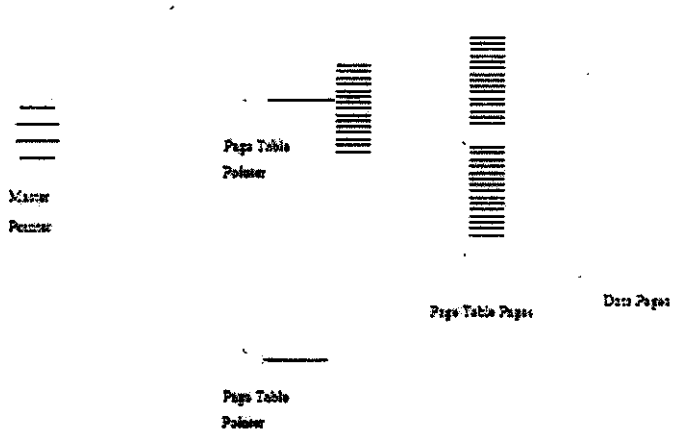
	and these modifications will not affect other copies." (page 12). This is further illustrated in Ylonen Fig.3:  Fig. 3 teaches that Snapshot 9 is an image of active database version Current State 1 at a past consistency point, while Snapshot 25 is an image of active database version Current State 2. Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
25. A method as in claim 24, wherein each snapshot includes a complete hierarchy for file system data, separate and apart from active file system data for the plural active file systems.	Ylonen Fig. 5 and the text in section 3.5 teach that each permanent snapshot includes a hierarchy of system data, such as Page Table Pointer and Page Table Pages. Some of these permanent snapshots are active database versions.

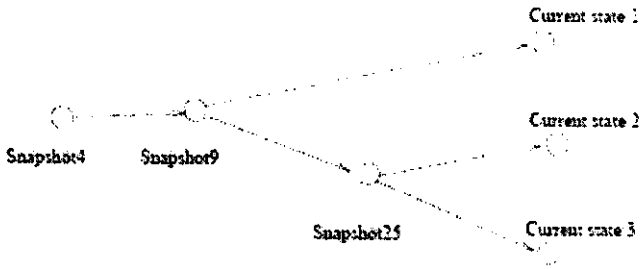
	 Figure 5: Shadow paging file structure with a master pointer and multiple versions. Furthermore, Hitz Figs. 3 and 4 at p. 11 teach that an active file system and snapshots have separate file system data hierarchies. Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
26. A memory as in claim 24, wherein at least one of the snapshots is converted into a new active file system.	Ylonen Fig. 3 teaches snapshots converted to active database versions.  Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-

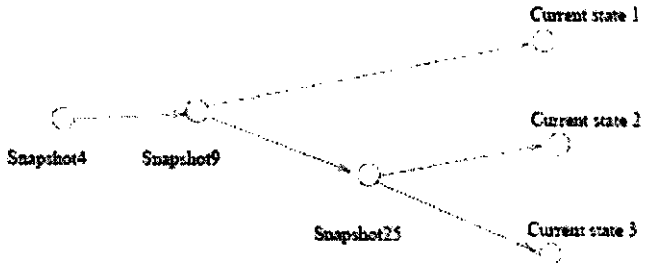
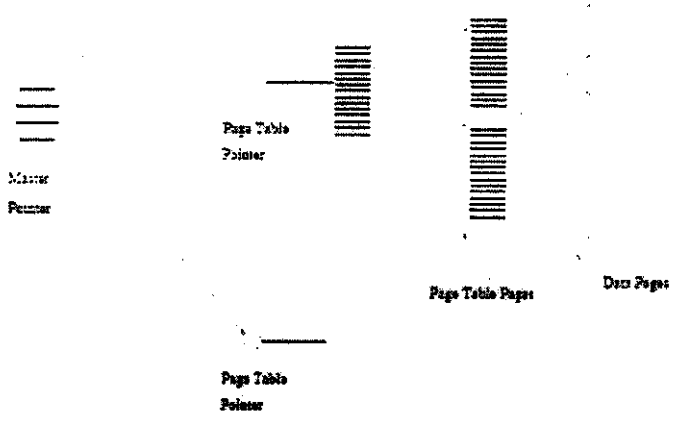
	write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
27. A memory as in claim 26, wherein the one of the snapshots is converted by making the one of the snapshots writable.	Ylonen section 3.4 teaches allowing modifications to certain snapshots so as to covert them to writable versions, as shown in Fig. 3. Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
28. A memory as in claim 27, wherein snapshot pointers from any of the active file systems to the new active file system are severed.	Ylonen section 3.4 and Fig. 3 teach that active database versions are independent and unsynchronized, diverging over time. Fig. 3 further shows that there are no snapshot pointers between any of the active database versions. Snapshot pointers point only from snapshots to their respective active database versions.  Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-

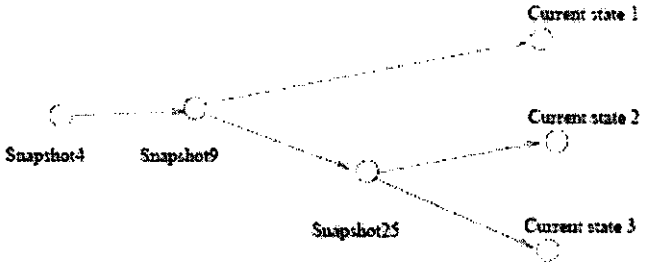
	write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
29. A memory storing information including instructions, the instructions executable by a processor to create plural active file systems, the instructions comprising the steps of: making a snapshot of a first active file system, the snapshot initially sharing data with the first active file system; and converting the snapshot to a second active file system by making the snapshot writable, with changes made to the first active file system not reflected in the second active file system, and with changes made to the second active file system not reflected in the first active file system.	Ylonen section 3 teaches maintaining snapshots that are a transaction-consistent copy of a database through shadow paging techniques. Fig. 2 in Ylonen section 3.1 further illustrates maintaining multiple snapshots. Ylonen section 3.4 teaches that such snapshots are a copy-on-write copy of the database. "From the user's point of view, each snapshot is an independent copy of the database. It is possible to make modifications to the copy, and these modifications will not affect other copies." (page 12). This is further illustrated in Ylonen Fig.3:  Thus, the writable snapshots of Ylonen initially access shared data and grow apart over time. For example, Current State 3 and Current State 2 in Fig. 3 above initially access data of parent Snapshot 25. Ylonen section 3.4 teaches that "copies diverge as more modifications are made." Ylonen further teaches in section 2, (page 5) that the pages of data are stored on disks operable under the control of a CPU, which inherently operates the data storage device by executing instructions. Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file

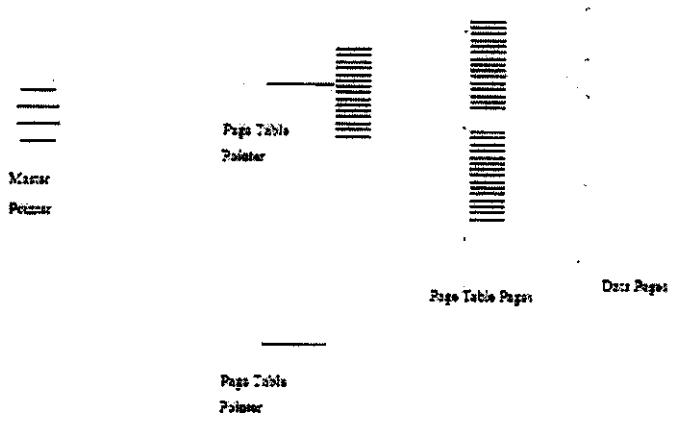
	system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
30. A memory as in claim 29, wherein when changes are made to the first active file system, modified data is recorded in the first active file system in a location that is not shared with the second active file system.	Ylonen section 3.4 teaches that "From the user's point of view, each snapshot is an independent copy of the database. It is possible to make modifications to the copy, and these modifications will not affect other copies." (page 12). Ylonen Fig. 5 and the text in section 3.5 further teach that active database versions do not share changed data locations.  Figure 5: Shadow paging file structure with a master pointer and multiple versions. In Fig. 5 each Page Table Pointer represents permanent snapshots, some or all of which are active versions of the database. As taught in Ylonen section 2, the shadow paging technique (also known as copy-on-write) requires that changes in Data Pages are recorded in Page Table Pages. As database versions disclosed in Ylonen Fig. 5 diverge over time the respective Page Table Pages are not shared. In addition, changed Data Pages are recorded in different locations using shadow paging / copy-on-write techniques. Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file

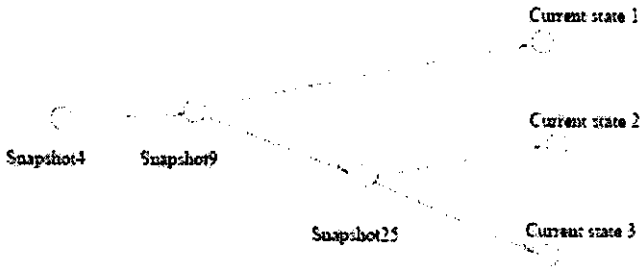
	system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
31. A memory as in claim 29, wherein when changes are made to the second active file system, modified data is recorded in the second active file system in a location that is not shared with the first active file system.	Ylonen section 3.4 teaches that "From the user's point of view, each snapshot is an independent copy of the database. It is possible to make modifications to the copy, and these modifications will not affect other copies." (page 12). Ylonen Fig. 5 and the text in section 3.5 further teach that active database versions do not share changed data locations.  Figure 5: Shadow paging file structure with a master pointer and multiple versions. In Fig. 5 each Page Table Pointer represents permanent snapshots, some or all of which are active versions of the database. As taught in Ylonen section 2, the shadow paging technique (also known as copy-on-write) requires that changes in Data Pages are recorded in Page Table Pages. As database versions disclosed in Ylonen Fig. 5 diverge over time the respective Page Table Pages are not shared. In addition, changed Data Pages are recorded in different locations using shadow paging / copy-on-write techniques. Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file

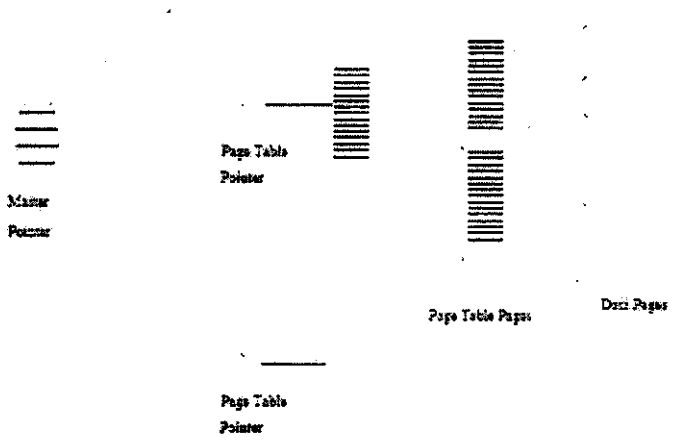
	system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
32. A memory as in claim 29, wherein the instructions further comprise the step of severing any snapshot pointers from the first active file system to the second active file system.	Ylonen section 3.4 and Fig. 3 teach that active database versions are independent and unsynchronized, diverging over time. Fig. 3 further shows that there are no snapshot pointers between any of the active database versions. Snapshot pointers point only from snapshots to their respective active database versions.  Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz..
33. A memory as in claim 29, wherein the instructions further comprise the steps of making snapshots of ones of the plural active file systems.	Fig. 3 of Ylonen teaches that Snapshot 9 is a snapshot of active database version Current State 1 while Snapshot 25 is a snapshot of active database version Current State 2.

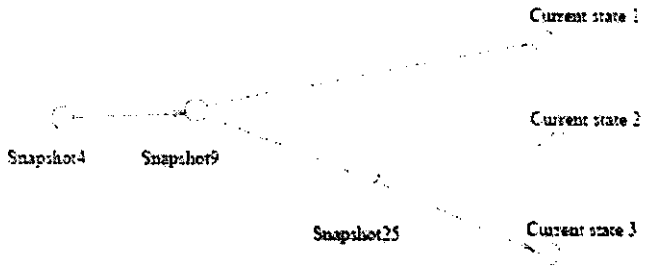
	 Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
34. A memory as in claim 33, wherein each snapshot includes a complete hierarchy for file system data, separate and apart from active file system data for the plural active file systems.	Ylonen Fig. 5 and the text in section 3.5 teach that each permanent snapshot includes a hierarchy of system data, such as Page Table Pointer and Page Table Pages. Some of these permanent snapshots are active database versions.  Figure 5: Shadow paging file structure with a master pointer and multiple versions. Furthermore, Hitz Figs. 3 and 4 at p. 11 teach that an active file system and snapshots have separate file system data hierarchies.

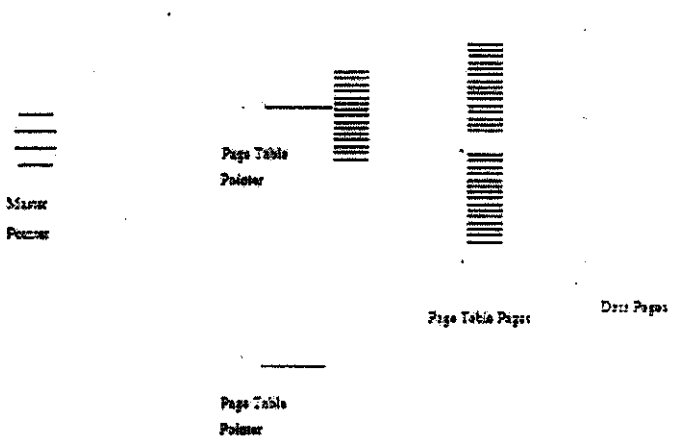
	Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
35. A memory as in claim 29, wherein the instructions further comprise the steps of: making a new snapshot of the first active file system, the new snapshot initially sharing data with the first active file system; converting the new snapshot to a third active file system by making the new snapshot writable, with changes made to the first active file system or the second active file system not reflected in the third active file system.	Ylonen Fig. 3 teaches making Snapshot 25 that initially shares data with active database version Current State 1 as both share the same parent Snapshot 9. Snapshot 25 is converted to active database version Current State 2. Changes made in each of Current State 1, Current State 2 and Current State 3 are not reflected in the other active database versions.  Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
36. A memory as in claim 35, wherein when changes are made to the first active file system or	Ylonen section 3.4 teaches that "From the user's point of view, each snapshot is an independent copy of the database. It is possible to make modifications to the copy, and these

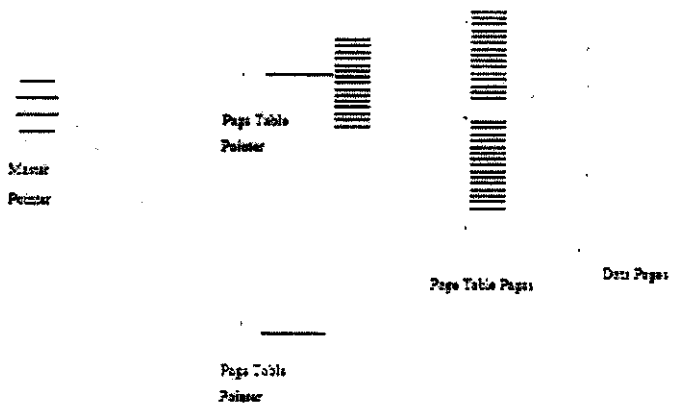
the second active file system, modified data is recorded in a location that is not shared with the third active file system.	modifications will not affect other copies." (page 12). Ylonen Fig. 5 and the text in section 3.5 further teach that active database versions do not share changed data locations.  Figure 5: Shadow paging file structure with a master pointer and multiple versions. In Fig. 5 each Page Table Pointer represents permanent snapshots, some or all of which are active versions of the database. As taught in Ylonen section 2, the shadow paging technique (also known as copy-on-write) requires that changes in Data Pages are recorded in Page Table Pages. As database versions disclosed in Ylonen Fig. 5 diverge over time the respective Page Table Pages are not shared. In addition, changed Data Pages are recorded in different locations using shadow paging / copy-on-write techniques. Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
37. A memory as in claim 29, wherein the instructions further comprise the steps of: making a	Ylonen Fig. 3 teaches making Snapshot 25 that initially shares data with active database version Current State 1 as both share the same parent Snapshot 9. Snapshot 25 is

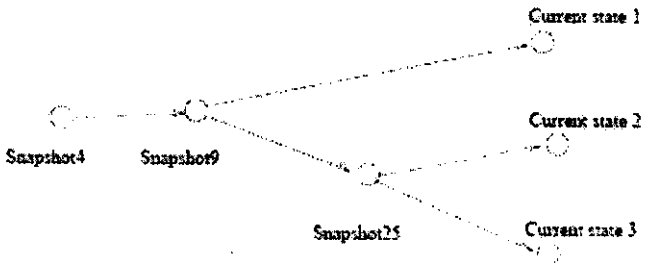
new snapshot of the second active file system, the new snapshot initially sharing data with the second active file system; converting the new snapshot to a third active file system by making the new snapshot writable, with changes made to the first active file system or the second active file system not reflected in the third active file system.	converted to active database version Current State 2. Changes made in each of Current State 1, Current State 2 and Current State 3 are not reflected in the other active database versions.  Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
38. A memory as in claim 37, wherein when changes are made to the first active file system or the second active file system, modified data is recorded in a location that is not shared with the third active file system.	Ylonen section 3.4 teaches that "From the user's point of view, each snapshot is an independent copy of the database. It is possible to make modifications to the copy, and these modifications will not affect other copies." (page 12). Ylonen Fig. 5 and the text in section 3.5 further teach that active database versions do not share changed data locations.

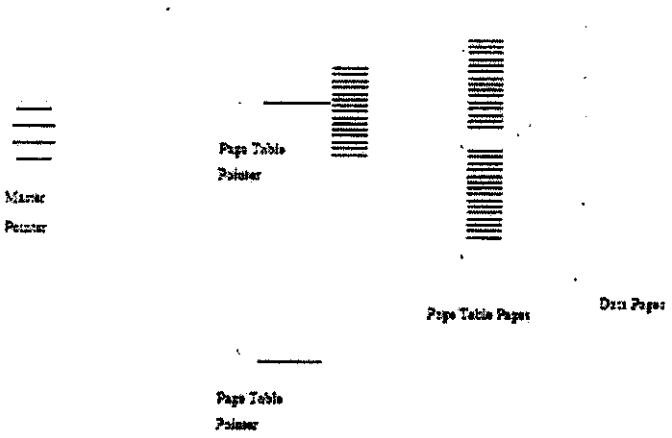
	 Figure 5: Shadow paging file structure with a master pointer and multiple versions. In Fig. 5 each Page Table Pointer represents permanent snapshots, some or all of which are active versions of the database. As taught in Ylonen section 2, the shadow paging technique (also known as copy-on-write) requires that changes in Data Pages are recorded in Page Table Pages. As database versions disclosed in Ylonen Fig. 5 diverge over time the respective Page Table Pages are not shared. In addition, changed Data Pages are recorded in different locations using shadow paging / copy-on-write techniques. Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
39. A storage system, comprising: at least one storage device; an interface to at least one computing device or network for receiving and sending information; and a controller that controls storage	Hitz at p. 4 teaches a file server appliance interfacing to a network. The disclosed file server inherently has a controller that operates under WAFL file system program control to store and retrieve information. Hitz at p. 4 further teaches that WAFL uses copy-on-write technique to implement snapshots. Hitz at p. 5 teaches that the file server appliance and WAFL support RAID (Redundant Array of Independent

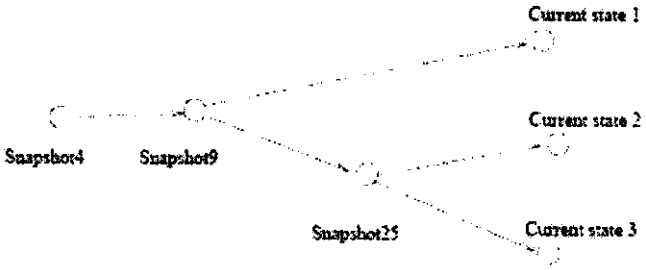
and retrieval of the information in the storage device, the controller operating under program control to maintain plural active file systems, wherein each of the active file systems initially access data shared with another of the active file systems, and wherein changes made to each of the active file systems are not reflected in the active file system with which the changed active file system shares the data.	Disks), constituting at least one storage device. Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz. Ylonen section 3 teaches maintaining snapshots that are a transaction-consistent copy of a database through shadow paging techniques. Fig. 2 in Ylonen section 3.1 further illustrates maintaining multiple snapshots. Ylonen section 3.4 teaches that such snapshots are a copy-on-write copy of the database. "From the user's point of view, each snapshot is an independent copy of the database. It is possible to make modifications to the copy, and these modifications will not affect other copies." (page 12). This is further illustrated in Ylonen Fig.3:  Thus, the writable snapshots of Ylonen initially access shared data and grow apart over time for database can be used in file systems. For example, Current State 3 and Current State 2 in Fig. 3 above initially access data of parent Snapshot 25. Ylonen section 3.4 teaches that "copies diverge as more modifications are made."
40. A storage system as in claim 39, wherein when a second active file system is created based on a first active file	Ylonen teaches in section 3.4 and Fig. 3 that a second active database version (e.g. Current State 2 in Fig. 3) is created based on a first active database version (Current State 1), with the two versions initially sharing data (Snapshot 9).

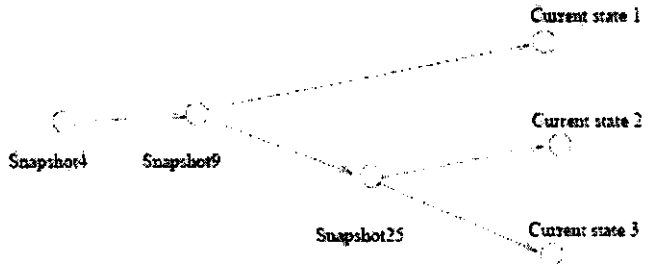
system, the first active file system and the second active file system initially share data.	Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
41. A storage system as in claim 40, wherein when changes are made to the first active file system, modified data is recorded in the first active file system in a location that is not shared with the second active file system.	Ylonen section 3.4 teaches that "From the user's point of view, each snapshot is an independent copy of the database. It is possible to make modifications to the copy, and these modifications will not affect other copies." (page 12). Ylonen Fig. 5 and the text in section 3.5 further teach that active database versions do not share changed data locations.  Figure 5: Shadow paging file structure with a master pointer and multiple versions. In Fig. 5 each Page Table Pointer represents an active version of the database. As taught in Ylonen section 2, the shadow paging technique (also known as copy-on-write) requires that changes in Data Pages are recorded in Page Table Pages. As database versions disclosed in Ylonen Fig. 5 diverge over time the respective Page Table Pages are not shared. In addition, changed Data Pages are recorded in different locations using shadow paging / copy-on-write techniques.

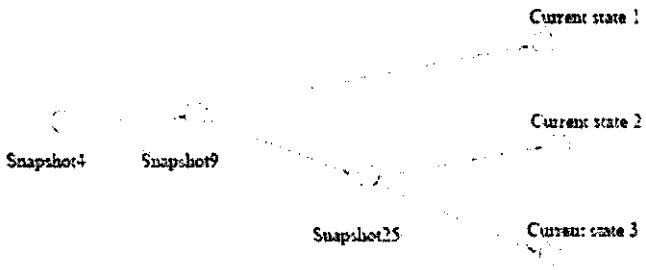
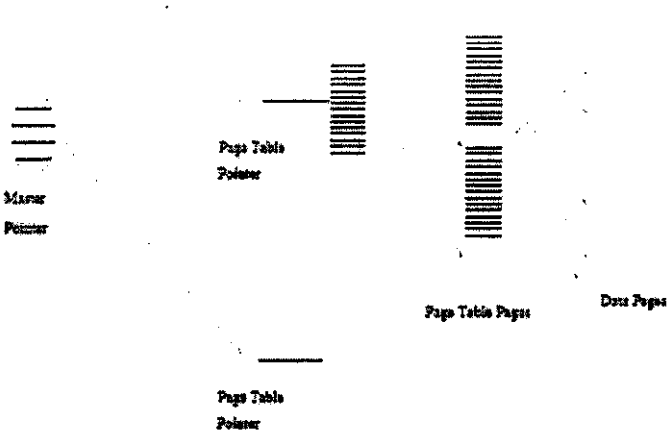
	Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
42. A storage system as in claim 40, wherein when changes are made to the second active file system, modified data is recorded in the second active file system in a location that is not shared with the first active file system.	Ylonen section 3.4 teaches that "From the user's point of view, each snapshot is an independent copy of the database. It is possible to make modifications to the copy, and these modifications will not affect other copies." Ylonen Fig. 5 and the text in section 3.5 further teach that active database versions do not share changed data locations.  Figure 5: Shadow paging file structure with a master pointer and multiple versions. In Fig. 5 each Page Table Pointer represents an active version of the database. As taught in Ylonen section 2, the shadow paging technique (also known as copy-on-write) requires that changes in Data Pages are recorded in Page Table Pages. As database versions disclosed in Ylonen Fig. 5 diverge over time the respective Page Table Pages are not shared. In addition, changed Data Pages are recorded in different locations using shadow paging / copy-on-write techniques. Author David Hitz, also one of the named inventors of the

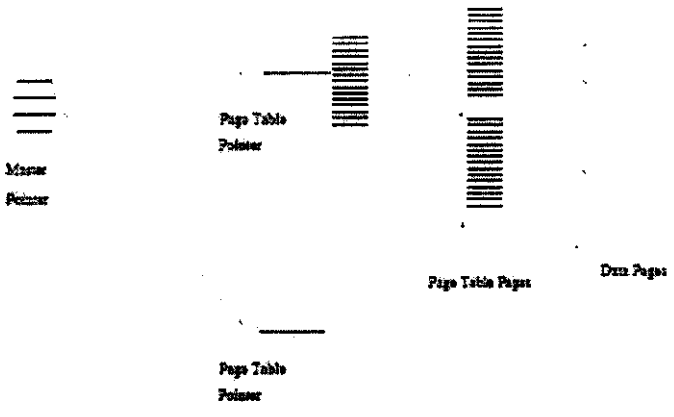
	'001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
43. A storage system as in claim 39, wherein snapshots are made of ones of the plural active file systems, each snapshot forming an image of its respective active file system at a past consistency point.	Ylonen section 3 teaches maintaining snapshots that are a transaction-consistent copy of a database through shadow paging techniques. Fig. 2 in Ylonen section 3.1 further illustrates maintaining multiple snapshots. Ylonen section 3.4 teaches modifying such snapshots that are a copy-on-write copy of the database. "From the user's point of view, each snapshot is an independent copy of the database. It is possible to make modifications to the copy, and these modifications will not affect other copies." (page 12). This is further illustrated in Ylonen Fig.3:  Fig. 3 teaches that Snapshot 9 is an image of active database version Current State 1 at a past consistency point, while Snapshot 25 is an image of active database version Current State 2. Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor.

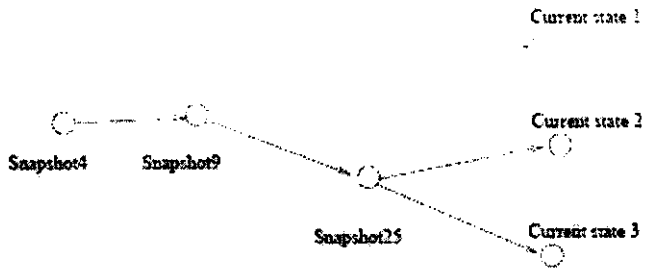
	Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
44. A storage system as in claim 43, wherein each snapshot includes a complete hierarchy for file system data, separate and apart from active file system data for the plural active file systems.	Ylonen Fig. 5 and the text in section 3.5 teach that each permanent snapshot includes a hierarchy of system data, such as Page Table Pointer and Page Table Pages. Some of these permanent snapshots are active database versions.  Figure 5: Shadow paging file structure with a master pointer and multiple versions. Furthermore, Hitz Figs. 3 and 4 at p. 11 teach that an active file system and snapshots have separate file system data hierarchies. Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
45. A storage system as in claim 43, wherein at least one of the snapshots is converted into a new active file system.	Ylonen Fig. 3 teaches snapshots converted to active database versions.

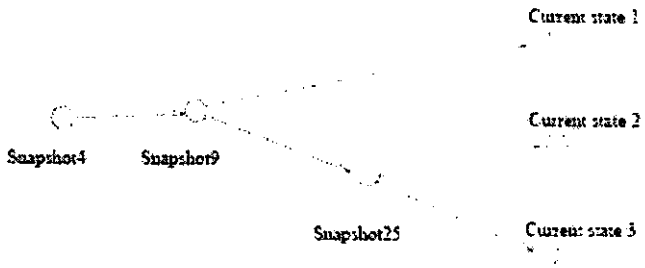
	 Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
46. A storage system as in claim 45, wherein the one of the snapshots is converted by making the one of the snapshots writable.	Ylonen section 3.4 teaches allowing modifications to certain snapshots so as to covert them to writable as shown in Fig. 3, and as discussed in section 3.4. Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
47. A storage system as in claim 46, wherein snapshot pointers from any of the active file systems to the new active file system are severed.	Ylonen section 3.4 and Fig. 3 teach that active database versions are independent and unsynchronized, diverging over time. Fig. 3 further shows that there are no snapshot pointers between any of the active database versions. Snapshot pointers point only from snapshots to their respective active database versions.

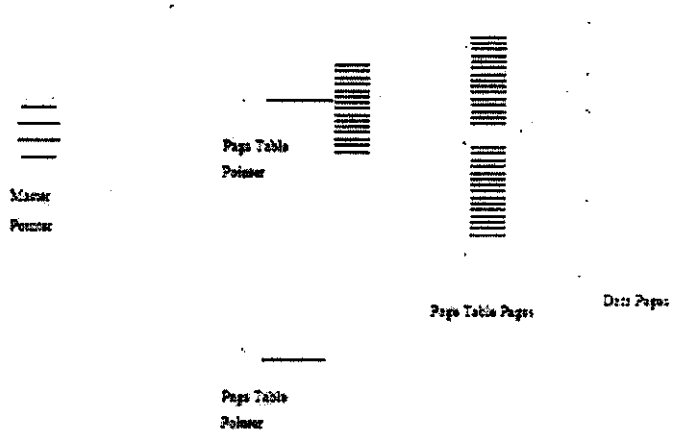
	 Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
48. A storage system, comprising: at least one storage device; an interface to at least one computing device or network for receiving and sending information; and a controller that controls storage and retrieval of the information in the storage device, the controller operating under program control to create plural active file systems, the program control comprising the steps of: making a snapshot of a first active file system, the snapshot initially sharing data with the first active file system; and converting the snapshot to a second active file system by making the snapshot writable, with changes made to the first active file system not reflected in the second active file system, and with changes made to the second active file system not	Hitz at p. 4 teaches a file server appliance interfacing to a network. The disclosed file server inherently has a controller that operates under WAFL file system program control to store and retrieve information. Hitz at p. 4 further teaches that WAFL uses copy-on-write technique to implement snapshots. Hitz at p. 5 teaches that the file server appliance and WAFL support RAID (Redundant Array of Independent Disks), constituting at least one storage device. Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz. Ylonen section 3 teaches maintaining snapshots that are a transaction-consistent copy of a database through shadow

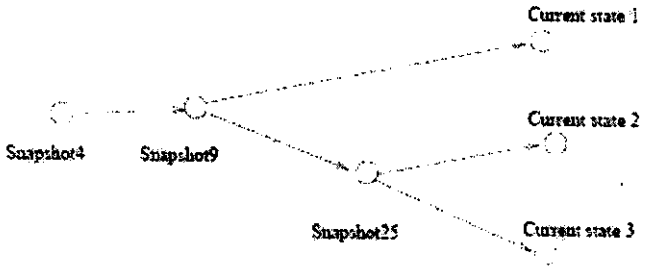
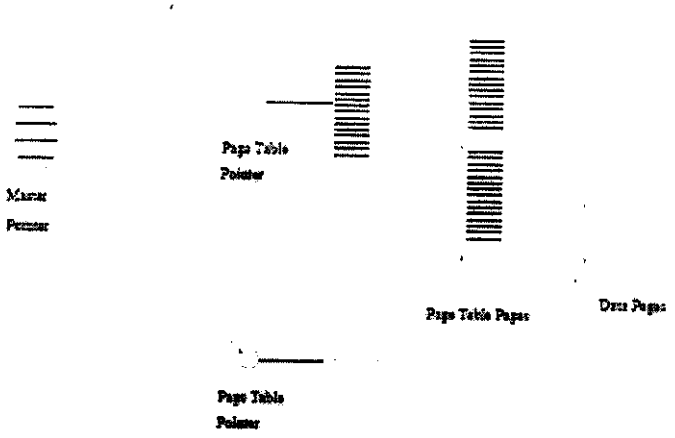
reflected in the first active file system.	paging techniques. Fig. 2 in Ylonen section 3.1 further illustrates maintaining multiple snapshots. Ylonen section 3.4 teaches that such snapshots are a copy-on-write copy of the database. "From the user's point of view, each snapshot is an independent copy of the database. It is possible to make modifications to the copy, and these modifications will not affect other copies." This is further illustrated in Ylonen Fig.3:  Thus, as would be readily appreciated by one of ordinary skill in the art, the writable snapshots of Ylonen initially access shared data and grow apart over time. For example, Current State 3 and Current State 2 in Fig. 3 above initially access data of parent Snapshot 25. Ylonen section 3.4 teaches that "copies diverge as more modifications are made."
49. A storage system as in claim 48, wherein when changes are made to the first active file system, modified data is recorded in the first active file system in a location that is not shared with the second active file system.	Ylonen section 3.4 teaches that "From the user's point of view, each snapshot is an independent copy of the database. It is possible to make modifications to the copy, and these modifications will not affect other copies." (page 12). Ylonen Fig. 5 and the text in section 3.5 further teach that active database versions do not share changed data locations.  Figure 5: Shadow paging file structure with a master pointer and multiple versions.

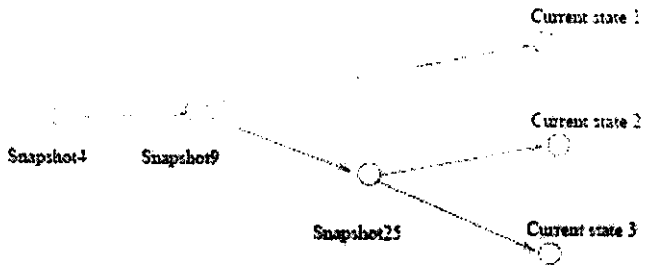
	In Fig. 5 each Page Table Pointer represents permanent snapshots, some or all of which are active versions of the database. As taught in Ylonen section 2, the shadow paging technique (also known as copy-on-write) requires that changes in Data Pages are recorded in Page Table Pages. As database versions disclosed in Ylonen Fig. 5 diverge over time the respective Page Table Pages are not shared. In addition, changed Data Pages are recorded in different locations using shadow paging / copy-on-write techniques. Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
50. A storage system as in claim 48, wherein when changes are made to the second active file system, modified data is recorded in the second active file system in a location that is not shared with the first active file system.	Ylonen section 3.4 teaches that "From the user's point of view, each snapshot is an independent copy of the database. It is possible to make modifications to the copy, and these modifications will not affect other copies." (page 12). Ylonen Fig. 5 and the text in section 3.5 further teach that active database versions do not share changed data locations.  Figure 5: Shadow paging file structure with a master pointer and multiple versions.

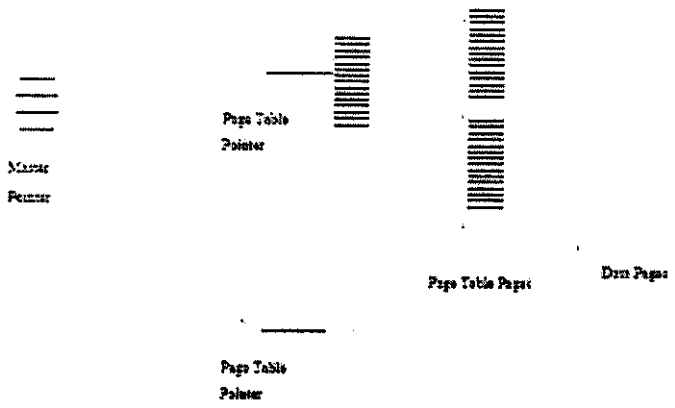
	In Fig. 5 each Page Table Pointer represents permanent snapshots, some or all of which are active versions of the database. As taught in Ylonen section 2, the shadow paging technique (also known as copy-on-write) requires that changes in Data Pages are recorded in Page Table Pages. As database versions disclosed in Ylonen Fig. 5 diverge over time the respective Page Table Pages are not shared. In addition, changed Data Pages are recorded in different locations using shadow paging / copy-on-write techniques. Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
51. A storage system as in claim 48, wherein the program control further comprises the step of severing any snapshot pointers from the first active file system to the second active file system.	Ylonen section 3.4 and Fig. 3 teach that active database versions are independent and unsynchronized, diverging over time. Fig. 3 further shows that there are no snapshot pointers between any of the active database versions. Snapshot pointers point only from snapshots to their respective active database versions.  Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of

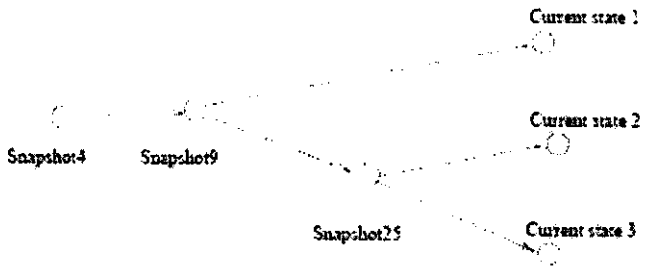
	the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
52. A storage system as in claim 48, wherein the program control further comprises the steps of making snapshots of ones of the plural active file systems.	Fig. 3 teaches that Snapshot 9 is a snapshot of active database version Current State 1 while Snapshot 25 is a snapshot of active database version Current State 2.  Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
53. A storage system as in claim 52, wherein each snapshot includes a complete hierarchy for file system data, separate and apart from active file system data for the plural active file systems.	Ylonen Fig. 5 and the text in section 3.5 teach that each permanent snapshot includes a hierarchy of system data, such as Page Table Pointer and Page Table Pages. Some of these permanent snapshots are active database versions.

	 Figure 5: Shadow paging file structure with a master pointer and multiple versions. Furthermore, Hitz Figs. 3 and 4 at p. 11 teach that an active file system and snapshots have separate file system data hierarchies. Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
54. A storage system as in claim 48, wherein the program control further comprises the steps of: making a new snapshot of the first active file system, the new snapshot initially sharing data with the first active file system; converting the new snapshot to a third active file system by making the new snapshot writable, with changes made to the first active file system or the second active file system not	Ylonen Fig. 3 teaches making Snapshot 25 that initially shares data with active database version Current State 1 as both share the same parent Snapshot 9. Snapshot 25 is converted to active database version Current State 2. Changes made in each of Current State 1, Current State 2 and Current State 3 are not reflected in the other active database versions.

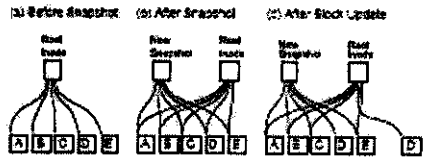
reflected in the third active file system.	 Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
55. A storage system as in claim 54, wherein when changes are made to the first active file system or the second active file system, modified data is recorded in a location that is not shared with the third active file system.	Ylonen section 3.4 teaches that "From the user's point of view, each snapshot is an independent copy of the database. It is possible to make modifications to the copy, and these modifications will not affect other copies." Ylonen Fig. 5 and the text in section 3.5 further teach that active database versions do not share changed data locations.  Figure 5: Shadow paging file structure with a master pointer and multiple versions. In Fig. 5 each Page Table Pointer represents permanent snapshots, some or all of which are active versions of the

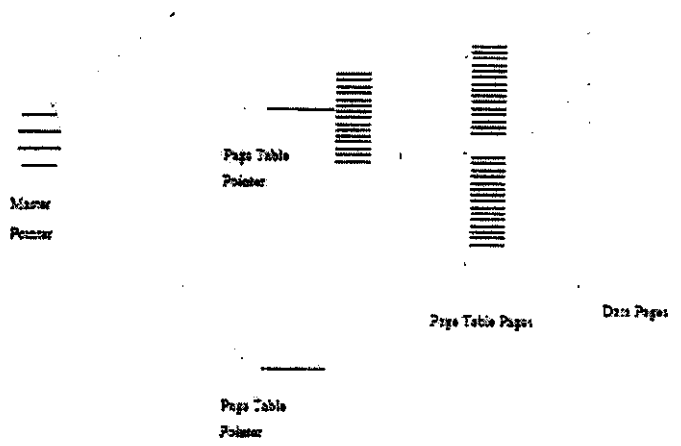
	database. As taught in Ylonen section 2, the shadow paging technique (also known as copy-on-write) requires that changes in Data Pages are recorded in Page Table Pages. As database versions disclosed in Ylonen Fig. 5 diverge over time the respective Page Table Pages are not shared. In addition, changed Data Pages are recorded in different locations using shadow paging / copy-on-write techniques. Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
56. A storage system as in claim 48, wherein the program control further comprises the steps of: making a new snapshot of the second active file system, the new snapshot initially sharing data with the second active file system; converting the new snapshot to a third active file system by making the new snapshot writable, with changes made to the first active file system or the second active file system not reflected in the third active file system.	Ylonen Fig. 3 teaches making Snapshot 25 that initially shares data with active database version Current State 1 as both share the same parent Snapshot 9. Snapshot 25 is converted to active database version Current State 2. Changes made in each of Current State 1, Current State 2 and Current State 3 are not reflected in the other active database versions.  Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor.

	Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
57. A storage system as in claim 56, wherein when changes are made to the first active file system or the second active file system, modified data is recorded in a location that is not shared with the third active file system.	Ylonen section 3.4 teaches that "From the user's point of view, each snapshot is an independent copy of the database. It is possible to make modifications to the copy, and these modifications will not affect other copies." Ylonen Fig. 5 and the text in section 3.5 further teach that active database versions do not share changed data locations.  Figure 5: Shadow paging file structure with a master pointer and multiple versions. In Fig. 5 each Page Table Pointer represents permanent snapshots, some or all of which are active versions of the database. As taught in Ylonen section 2, the shadow paging technique (also known as copy-on-write) requires that changes in Data Pages are recorded in Page Table Pages. As database versions disclosed in Ylonen Fig. 5 diverge over time the respective Page Table Pages are not shared. In addition, changed Data Pages are recorded in different locations using shadow paging / copy-on-write techniques. Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor.

	Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
58. An apparatus for operating data storage, the apparatus including means for creating plural active file systems and means for maintaining plural active file systems, wherein each of the active file systems initially access data shared with another of the active file systems, and wherein changes made to each of the active file systems are not reflected in the active file system with which the changed active file system shares the data.	Ylonen section 3 teaches maintaining snapshots that are a transaction-consistent copy of a database through shadow paging techniques. Fig. 2 in Ylonen section 3.1 further illustrates maintaining multiple snapshots. Ylonen section 3.4 teaches that such snapshots are a copy-on-write copy of the database. "From the user's point of view, each snapshot is an independent copy of the database. It is possible to make modifications to the copy, and these modifications will not affect other copies." This is further illustrated in Ylonen Fig.3:  Thus, the writable snapshots of Ylonen initially access shared data and grow apart over time. For example, Current State 3 and Current State 2 in Fig. 3 above initially access data of parent Snapshot 25. Ylonen section 3.4 teaches that "copies diverge as more modifications are made." Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
59. An apparatus for creating plural active file systems, comprising: means for making a	Ylonen section 3 teaches maintaining snapshots that are a transaction-consistent copy of a database through shadow paging techniques. Fig. 2 in Ylonen section 3.1 further

snapshot of a first active file system, the snapshot initially sharing data with the first active file system; and means for converting the snapshot to a second active file system by making the snapshot writable, with changes made to the first active file system not reflected in the second active file system, and with changes made to the second active file system not reflected in the first active file system.	illustrates maintaining multiple snapshots. Ylonen section 3.4 teaches that such snapshots are a copy-on-write copy of the database. "From the user's point of view, each snapshot is an independent copy of the database. It is possible to make modifications to the copy, and these modifications will not affect other copies." This is further illustrated in Ylonen Fig.3:  Thus, the writable snapshots of Ylonen initially access shared data and grow apart over time. For example, Current State 3 and Current State 2 in Fig. 3 above initially access data of parent Snapshot 25. Ylonen section 3.4 teaches that "copies diverge as more modifications are made." Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
60. A method of operating data storage, comprising: making a snapshot of organizational data of a first active file system, the snapshot pointing to original non-organizational data of the first active file system; storing the snapshot; modifying a first portion of the original non-organizational data of the first active file system in response to	Hitz section 3.4 and Figs. 3 and 4 teach taking a snapshot of a root inode (file system organizational data), the root inode pointing to original data blocks (non-organizational data, labeled as A-E in Fig. 3). When original data blocks are modified (block D' in Fig. 3(c)) in response to a WAFL write operation, the modified data blocks become part of modified data blocks of the file system, as shown in Fig. 3(c). The modified block D' are written so as not to overwrite the original block D, which is still pointed to by the snapshot (*see* Hitz p. 10). As further shown in Figs. 3(b) and 3(c), the snapshot root inode points to original data blocks, while the

a first active file system access request, resulting in a modified first portion being part of first modified non-organizational data of the first active file system; and storing the modified first portion so as not to overwrite the first portion; wherein, after the step of storing the modified first portion, the snapshot points to the original non-organizational data, the organizational data of the first active file system point to the first modified non-organizational data of the first filing system, and the original non-organizational data and the first modified non-organizational data partially overlap.	active file system inode points to modified data blocks, with the two sets of data blocks partially overlapping:  Figure 3 WAFL creates a Snapshot by duplicating the root inode that describes the inode file. WAFL avoids changing blocks in a Snapshot by writing new data to new locations on disk. Ylonen sections 2 and 3 and Fig. 1 teach a shadow paging method of snapshotting that is equivalent to the copy-on-write snapshots taught by Hitz and described above. Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
61. A method according to claim 60, wherein: the step of storing the snapshot comprises storing the snapshot as a second active filing system; the method further comprising: modifying a second portion of the original non-organizational data in response to a second active file system access request, resulting in a modified second portion being part of second modified non-organizational data of the second active file system; and storing the modified second portion so as not to overwrite the second portion; wherein,	Ylonen section 3.4 and Fig. 3 teach that a snapshot can be converted to a second active database version. Ylonen section 3.4 teaches that "From the user's point of view, each snapshot is an independent copy of the database. It is possible to make modifications to the copy, and these modifications will not affect other copies." (page 12). Ylonen Fig. 5 and the text in section 3.5 further teach that active database versions do not share changed data locations.

after the step of storing the modified second portion, the snapshot points to the second modified non-organizational data, the organizational data of the first active file system point to the first modified non-organizational data, and the first modified non-organizational data and the second modified non-organizational data partially overlap.	 Figure 5: Shadow paging file structure with a master pointer and multiple versions. In Fig. 5 each Page Table Pointer represents an active version of the database. As taught in Ylonen section 2, the shadow paging technique (also known as copy-on-write) requires that changes in Data Pages are recorded in Page Table Pages. As database versions disclosed in Ylonen Fig. 5 diverge over time the respective Page Table Pages are not shared. In addition, changed Data Pages are recorded in different locations using shadow paging / copy-on-write techniques. Thus, once a snapshot becomes an active database version, its organizational data (table pointers and page tables) point to modified data pages (non-organizational data). These data pages partially overlap with those of other active database versions, but diverge over time. Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
62. A method according to claim 61, wherein the step of making a snapshot is performed at a	Ylonen section 3 at p. 7 teaches that "[a] snapshot is a transaction-consistent copy of the database." Thus, transaction consistency represents a consistency point.

consistency point of the first active file system.	Hitz section 3.5 at p. 12 further discloses taking snapshots at consistency points. Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.
63. A method according to claim 62, wherein data is stored in the first and second active file systems using blocks.	Hitz in section 3 teaches storing snapshot data using blocks. Author David Hitz, also one of the named inventors of the '001 Patent, admits in section 3.5 (page 12) that copy-on-write snapshot techniques implemented in the WAFL file system, the preferred embodiment of the '001 Patent, are well known for databases. Thus, as a matter of claim construction, one of ordinary skill in the art would understand that "file system," as claimed in the '001 patent, includes "databases," and the claim is anticipated by Ylonen with the admission of the named inventor. Alternatively, one of ordinary skill in the art would readily appreciate that the database snapshots taught in Ylonen can be implemented with file system snapshots taught in Hitz.

Second Basis of Invalidity

The reference applicable to the second basis of invalidity is:

1. Veritas File System 3.4 Administrator's Guide, November 2000 ("VxFS").

The pertinence and manner of applying VxFS to claims 1-63 for which re-examination is requested is as follows:

Claims of '001 Patent	VxFS
1. A method of operating data	VxFS discloses a file system. Chapter 8 discloses storage

storage, the method including maintenance of plural active file systems, wherein each of the active file systems initially access data shared with another of the active file systems, and wherein changes made to each of the active file systems are not reflected in the active file system with which the changed active file system shares the data.	checkpoints, a copy-on-write snapshot of the file system. VxFS storage checkpoints can be mounted as writable checkpoints using the –o rw or –o remount options. (*See* p. 86, p. 89.) When such a storage checkpoint is mounted as writable, it initially shares data with the original file system through copy-on-write techniques. For instance, at pp. 84-85, VxFS teaches that "[t]he Storage Checkpoint presents the exact image of the file system by finding the data from the primary fileset. . . . This is called the *copy-on-write* technique, which allows the Storage Checkpoint to preserve the image of the primary fileset when the Storage Checkpoint is taken." Once a storage checkpoint is mounted as writable, it is inherent that it becomes an independent active file system. For instance, VxFS at p. 86 states that "[y]ou can mount, access, and write to a data Storage Checkpoint just as you would to a file system." Furthermore, VxFS at p. 89 warns that such a writable checkpoint will diverge from a read-only storage checkpoint when it is written to. Changes made to a writable storage checkpoint are not reflected in the original file system.
2. A method as in claim 1, wherein when a second active file system is created based on a first active file system, the first active file system and the second active file system initially share data.	VxFS at pp. 83-85 and elsewhere discloses that storage checkpoints are made using copy-on-write techniques. When such a storage checkpoint is mounted as writable, it initially shares data with the original file system.
3. A method as in claim 2, wherein when changes are made to the first active file system, modified data is recorded in the first active file system in a location that is not shared with the second active file system.	Because VxFS storage checkpoints use copy-on-write techniques, as disclosed at pp. 83-85 and elsewhere, it is inherent that changes to either the writable storage checkpoint or the original file system are stored in different locations so as to not overwrite any live data.
4. A method as in claim 2, wherein when changes are made to the second active file system, modified data is recorded in the second active file system in a location that is not shared with the first active file system.	Because VxFS storage checkpoints use copy-on-write techniques, as disclosed at pp. 83-85 and elsewhere, it is inherent that changes to either the writable storage checkpoint or the original file system are stored in different locations so as to not overwrite any live data.
5. A method as in claim 1,	VxFS at p. 83 teaches that storage checkpoints can be taken

wherein snapshots are made of ones of the plural active file systems, each snapshot forming an image of its respective active file system at a past consistency point.	of multiple file systems. Thus, at p. 83 it is disclosed that [y]ou can create a Storage Checkpoint on a single file system or a list of file systems. A multiple file system Storage Checkpoint simultaneously freezes the file systems, creates a Storage Checkpoint on all file systems, and thaws the file systems." As disclosed at pp. 82-85, each checkpoint is an image of its file system at a past consistency point. Thus, at p. 83 it is disclosed that "[t]he storage checkpoint is logically identical to the primary fileset when the storage checkpoint is created." At pp. 84-85, VxFS teaches that "[t]he Storage Checkpoint presents the exact image of the file system by finding the data from the primary fileset. . . . This is called the *copy-on-write* technique, which allows the Storage Checkpoint to preserve the image of the primary fileset when the Storage Checkpoint is taken."
6. A method as in claim 5, wherein each snapshot includes a complete hierarchy for file system data, separate and apart from active file system data for the plural active file systems.	VxFS discloses on p. 86 and elsewhere that each storage snapshot contains file system metadata. For instance, as disclosed at pp. 83-85, this metadata includes pointers to data blocks. Appendix C further discloses that VxFS metadata includes a superblock, inodes and pointer tables for each file system.
7. A method as in claim 5, wherein at least one of the snapshots is converted into a new active file system.	VxFS discloses at pp. 83-85 and 89 that checkpoints can be taken of an active file system and that storage checkpoints can be mounted as writable, becoming other active file systems.
8. A method as in claim 7, wherein the one of the snapshots is converted by making the one of the snapshots writable.	VxFS discloses at pp. 83-85 and 89 that checkpoints can be taken of an active file system and that storage checkpoints can be mounted as writable, becoming other active file systems.
9. A method as in claim 8, wherein snapshot pointers from any of the active file systems to the new active file system are severed.	Once a storage checkpoint is mounted as writable, it is inherent that it becomes an independent active file system. For instance, VxFS at p. 86 states that "[y]ou can mount, access, and write to a data Storage Checkpoint just as you would to a file system." Furthermore, VxFS at p. 89 warns that such a writable checkpoint will diverge from a read-only storage checkpoint when it is written to. Therefore, no pointer exists between active file systems in VxFS.
10. A method of creating plural active file systems, comprising the steps of: making a snapshot of a	VxFS discloses a file system. Chapter 8 discloses storage checkpoints, a copy-on-write snapshot of the file system. VxFS storage checkpoints can be mounted as writable

first active file system, the snapshot initially sharing data with the first active file system; and converting the snapshot to a second active file system by making the snapshot writable, with changes made to the first active file system not reflected in the second active file system, and with changes made to the second active file system not reflected in the first active file system.	checkpoints using the –o rw or –o remount options. (*See* p. 86, p. 89.) When such a storage checkpoint is mounted as writable, it initially shares data with the original file system through copy-on-write techniques. For instance, at pp. 84-85, the VxFS teaches that "[t]he Storage Checkpoint presents the exact image of the file system by finding the data from the primary fileset. . . . This is called the *copy-on-write* technique, which allows the Storage Checkpoint to preserve the image of the primary fileset when the Storage Checkpoint is taken." Once a storage checkpoint is mounted as writable, it is inherent that it becomes an independent active file system. For instance, VxFS at p. 86 states that "[y]ou can mount, access, and write to a data Storage Checkpoint just as you would to a file system." Furthermore, VxFS at p. 89 warns that such a writable checkpoint will diverge from a read-only storage checkpoint when it is written to. Thus, changes made to a writable storage checkpoint are not reflected in the original file system.
11. A method as in claim 10, wherein when changes are made to the first active file system, modified data is recorded in the first active file system in a location that is not shared with the second active file system.	Because VxFS storage checkpoints use copy-on-write techniques, as disclosed at pp. 83-85 and elsewhere, it is inherent that changes to either the writable storage checkpoint or the original file system are stored in different locations so as to not overwrite any live data.
12. A method as in claim 10, wherein when changes are made to the second active file system, modified data is recorded in the second active file system in a location that is not shared with the first active file system.	Because VxFS storage checkpoints use copy-on-write techniques, as disclosed at pp. 83-85 and elsewhere, it is inherent that changes to either the writable storage checkpoint or the original file system are stored in different locations so as to not overwrite any live data.
13. A method as in claim 10, further comprising the step of severing any snapshot pointers from the first active file system to the second active file system.	Once a storage checkpoint is mounted as writable, it is inherent that it becomes an independent active file system. For instance, VxFS at p. 86 states that "[y]ou can mount, access, and write to a data Storage Checkpoint just as you would to a file system." Furthermore, VxFS at p. 89 warns that such a writable checkpoint will diverge from a read-only storage checkpoint when it is written to. Therefore, no pointer exists between active file systems in VxFS.
14. A method as in claim 10,	VxFS at p. 83 teaches that storage checkpoints can be taken

further comprising the steps of making snapshots of ones of the plural active file systems.	of multiple file systems. Thus, at p. 83 it is disclosed that [y]ou can create a Storage Checkpoint on a single file system or a list of file systems. A multiple file system Storage Checkpoint simultaneously freezes the file systems, creates a Storage Checkpoint on all file systems, and thaws the file systems." At pp. 84-85, VxFS teaches that "[t]he Storage Checkpoint presents the exact image of the file system by finding the data from the primary fileset. . . . This is called the *copy-on-write* technique, which allows the Storage Checkpoint to preserve the image of the primary fileset when the Storage Checkpoint is taken."
15. A method as in claim 14, wherein each snapshot includes a complete hierarchy for file system data, separate and apart from active file system data for the plural active file systems.	VxFS discloses on p. 86 and elsewhere that each storage snapshot contains file system metadata. For instance, as disclosed at pp. 83-85, this metadata includes pointers to data blocks. Appendix C further discloses that VxFS metadata includes a superblock, inodes and pointer tables for each file system.
16. A method as in claim 10, further comprising the steps of: making anew snapshot of the first active file system, the new snapshot initially sharing data with the first active file system; converting the new snapshot to a third active file system by making the new snapshot writable, with changes made to the first active file system or the second active file system not reflected in the third active file system.	VxFS storage checkpoints can be mounted as writable checkpoints using the –o rw or –o remount options. (*See* p. 86, p. 89.) When such a storage checkpoint is mounted as writable, it initially shares data with the original file system through copy-on-write techniques. Once a storage checkpoint is mounted as writable, it is inherent that it becomes an independent active file system. For instance, VxFS at p. 86 states that "[y]ou can mount, access, and write to a data Storage Checkpoint just as you would to a file system." Furthermore, VxFS at p. 89 warns that such a writable checkpoint will diverge from a read-only storage checkpoint when it is written to. Thus, changes made to a writable storage checkpoint are not reflected in the original file system. Once a VxFS storage checkpoint is mounted as an active file system, a storage snapshot can be taken of it.
17. A method as in claim 16, wherein when changes are made to the first active file system or the second active file system, modified data is recorded in a location that is not shared with the third active file system.	Because VxFS storage checkpoints use copy-on-write techniques, as disclosed at pp. 83-85 and elsewhere, it is inherent that changes to any newly mounted active file systems and changes to the "parent" file system are stored in different locations so as to not overwrite any live data.

18. A method as in claim 10, further comprising the steps of: making a new snapshot of the second active file system, the new snapshot initially sharing data with the second active file system; converting the new snapshot to a third active file system by making the new snapshot writable, with changes made to the first active file system or the second active file system not reflected in the third active file system.	VxFS storage checkpoints can be mounted as writable checkpoints using the –o rw or –o remount options. (*See* p. 86, p. 89.) When such a storage checkpoint is mounted as writable, it initially shares data with the original file system through copy-on-write techniques. Once a storage checkpoint is mounted as writable, it is inherent that it becomes an independent active file system. For instance, VxFS at p. 86 states that "[y]ou can mount, access, and write to a data Storage Checkpoint just as you would to a file system." Furthermore, VxFS at p. 89 warns that such a writable checkpoint will diverge from a read-only storage checkpoint when it is written to. Thus, changes made to a writable storage checkpoint are not reflected in the original file system. Once a VxFS storage checkpoint is mounted as an active file system, a storage snapshot can be taken of it.
19. A method as in claim 18, wherein when changes are made to the first active file system or the second active file system, modified data is recorded in a location that is not shared with the third active file system.	Because VxFS storage checkpoints use copy-on-write techniques, as disclosed at pp. 83-85 and elsewhere, it is inherent that changes to any newly mounted active file systems and changes to the "parent" file system are stored in different locations so as to not overwrite any live data.
20. A memory storing information including instructions, the instructions executable by a processor to operate data storage, the instructions comprising steps to maintain plural active file systems, wherein each of the active file systems initially access data shared with another of the active file systems, and wherein changes made to each of the active file systems are not reflected in the active file system with which the changed active file system shares the data.	VxFS discloses a file system. It is inherent that the file system is comprised of computer-executable instructions. Chapter 8 discloses storage checkpoints, a copy-on-write snapshot of the file system. VxFS storage checkpoints can be mounted as writable checkpoints using the –o rw or –o remount options. (*See* p. 86, p. 89.) When such a storage checkpoint is mounted as writable, it initially shares data with the original file system through copy-on-write techniques. For instance, at pp. 84-85, VxFS teaches that "[t]he Storage Checkpoint presents the exact image of the file system by finding the data from the primary fileset. . . . This is called the *copy-on-write* technique, which allows the Storage Checkpoint to preserve the image of the primary fileset when the Storage Checkpoint is taken." Once a storage checkpoint is mounted as writable, it is inherent that it becomes an independent active file system. For instance, VxFS at p. 86 states that "[y]ou can mount, access, and write to a data Storage Checkpoint just as you would to a file system." Furthermore, VxFS at p. 89 warns

	that such a writable checkpoint will diverge from a read-only storage checkpoint when it is written to. Thus, changes made to a writable storage checkpoint are not reflected in the original file system.
21. A memory as in claim 20, wherein when a second active file system is created based on a first active file system, the first active file system and the second active file system initially share data.	VxFS at pp. 83-85 and elsewhere discloses that storage checkpoints are made using copy-on-write techniques. When such a storage checkpoint is mounted as writable, it initially shares data with the original file system.
22. A memory as in claim 21, wherein when changes are made to the first active file system, modified data is recorded in the first active file system in a location that is not shared with the second active file system.	Because VxFS storage checkpoints use copy-on-write techniques, as disclosed at pp. 83-85 and elsewhere, it is inherent that changes to either the writable storage checkpoint or the original file system are stored in different locations so as to not overwrite any live data.
23. A memory as in claim 21, wherein when changes are made to the second active file system, modified data is recorded in the second active file system in a location that is not shared with the first active file system.	Because VxFS storage checkpoints use copy-on-write techniques, as disclosed at pp. 83-85 and elsewhere, it is inherent that changes to either the writable storage checkpoint or the original file system are stored in different locations so as to not overwrite any live data.
24. A memory as in claim 20, wherein snapshots are made of ones of the plural active file systems, each snapshot forming an image of its respective active file system at a past consistency point.	VxFS at p. 83 teaches that storage checkpoints can be taken of multiple file systems. Thus, at p. 83 it is disclosed that [y]ou can create a Storage Checkpoint on a single file system or a list of file systems. A multiple file system Storage Checkpoint simultaneously freezes the file systems, creates a Storage Checkpoint on all file systems, and thaws the file systems." As disclosed at pp. 82-85, each checkpoint is an image of its file system at a past consistency point. Thus, at p. 83 it is disclosed that "[t]he storage checkpoint is logically identical to the primary fileset when the storage checkpoint is created." At pp. 84-85, VxFS teaches that "[t]he Storage Checkpoint presents the exact image of the file system by finding the data from the primary fileset. . . . This is called the *copy-on-write* technique, which allows the Storage Checkpoint to preserve the image of the primary fileset when the Storage Checkpoint is taken."

25. A method as in claim 24, wherein each snapshot includes a complete hierarchy for file system data, separate and apart from active file system data for the plural active file systems.	VxFS discloses on p. 86 and elsewhere that each storage snapshot contains file system metadata. For instance, as disclosed at pp. 83-85, this metadata includes pointers to data blocks. Appendix C further discloses that VxFS metadata includes a superblock, inodes and pointer tables for each file system.
26. A memory as in claim 24, wherein at least one of the snapshots is converted into a new active file system.	VxFS discloses at pp. 83-85 and 89 that checkpoints can be taken of an active file system and that storage checkpoints can be mounted as writable, becoming other active file systems.
27. A memory as in claim 26, wherein the one of the snapshots is converted by making the one of the snapshots writable.	VxFS discloses at pp. 83-85 and 89 that checkpoints can be taken of an active file system and that storage checkpoints can be mounted as writable, becoming other active file systems.
28. A memory as in claim 27, wherein snapshot pointers from any of the active file systems to the new active file system are severed.	Once a storage checkpoint is mounted as writable, it is inherent that it becomes an independent active file system. For instance, VxFS at p. 86 states that "[y]ou can mount, access, and write to a data Storage Checkpoint just as you would to a file system." Furthermore, VxFS at p. 89 warns that such a writable checkpoint will diverge from a read-only storage checkpoint when it is written to. Therefore, no pointer exists between active file systems in VxFS.
29. A memory storing information including instructions, the instructions executable by a processor to create plural active file systems, the instructions comprising the steps of: making a snapshot of a first active file system, the snapshot initially sharing data with the first active file system; and converting the snapshot to a second active file system by making the snapshot writable, with changes made to the first active file system not reflected in the second active file system, and with changes made to the second active file system not reflected in the first active file system.	VxFS discloses a file system. It is inherent that the file system is comprised of computer-executable instructions. Chapter 8 discloses storage checkpoints, a copy-on-write snapshot of the file system. VxFS storage checkpoints can be mounted as writable checkpoints using the –o rw or –o remount options. (*See* p. 86, p. 89.) When such a storage checkpoint is mounted as writable, it initially shares data with the original file system through copy-on-write techniques. For instance, at pp. 84-85, VxFS teaches that "[t]he Storage Checkpoint presents the exact image of the file system by finding the data from the primary fileset. . . . This is called the *copy-on-write* technique, which allows the Storage Checkpoint to preserve the image of the primary fileset when the Storage Checkpoint is taken." Once a storage checkpoint is mounted as writable, it is inherent that it becomes an independent active file system. For instance, VxFS at p. 86 states that "[y]ou can mount, access, and write to a data Storage Checkpoint just as you would to a file system." Furthermore, VxFS at p. 89 warns that such a writable checkpoint will diverge from a read-

	only storage checkpoint when it is written to. Thus, changes made to a writable storage checkpoint are not reflected in the original file system.
30. A memory as in claim 29, wherein when changes are made to the first active file system, modified data is recorded in the first active file system in a location that is not shared with the second active file system.	Because VxFS storage checkpoints use copy-on-write techniques, as disclosed at pp. 83-85 and elsewhere, it is inherent that changes to either the writable storage checkpoint or the original file system are stored in different locations so as to not overwrite any live data.
31. A memory as in claim 29, wherein when changes are made to the second active file system, modified data is recorded in the second active file system in a location that is not shared with the first active file system.	Because VxFS storage checkpoints use copy-on-write techniques, as disclosed at pp. 83-85 and elsewhere, it is inherent that changes to either the writable storage checkpoint or the original file system are stored in different locations so as to not overwrite any live data.
32. A memory as in claim 29, wherein the instructions further comprise the step of severing any snapshot pointers from the first active file system to the second active file system.	Once a storage checkpoint is mounted as writable, it is inherent that it becomes an independent active file system. For instance, VxFS at p. 86 states that "[y]ou can mount, access, and write to a data Storage Checkpoint just as you would to a file system." Furthermore, VxFS at p. 89 warns that such a writable checkpoint will diverge from a read-only storage checkpoint when it is written to. Therefore, no pointer exists between active file systems in VxFS.
33. A memory as in claim 29, wherein the instructions further comprise the steps of making snapshots of ones of the plural active file systems.	VxFS at p. 83 teaches that storage checkpoints can be taken of multiple file systems. Thus, at p. 83 it is disclosed that [y]ou can create a Storage Checkpoint on a single file system or a list of file systems. A multiple file system Storage Checkpoint simultaneously freezes the file systems, creates a Storage Checkpoint on all file systems, and thaws the file systems." At pp. 84-85, VxFS teaches that "[t]he Storage Checkpoint presents the exact image of the file system by finding the data from the primary fileset. . . . This is called the *copy-on-write* technique, which allows the Storage Checkpoint to preserve the image of the primary fileset when the Storage Checkpoint is taken."
34. A memory as in claim 33, wherein each snapshot includes a complete hierarchy for file system data, separate and apart from	VxFS discloses on p. 86 and elsewhere that each storage snapshot contains file system metadata. For instance, as disclosed at pp. 83-85, this metadata includes pointers to data blocks. Appendix C further discloses that VxFS

active file system data for the plural active file systems.	metadata includes a superblock, inodes and pointer tables for each file system. The VxFS metadata is unique to each snapshot and to each active file system.
35. A memory as in claim 29, wherein the instructions further comprise the steps of: making a new snapshot of the first active file system, the new snapshot initially sharing data with the first active file system; converting the new snapshot to a third active file system by making the new snapshot writable, with changes made to the first active file system or the second active file system not reflected in the third active file system.	VxFS storage checkpoints can be mounted as writable checkpoints using the –o rw or –o remount options. (*See* p. 86, p. 89.) When such a storage checkpoint is mounted as writable, it initially shares data with the original file system through copy-on-write techniques. Once a storage checkpoint is mounted as writable, it is inherent that it becomes an independent active file system. For instance, VxFS at p. 86 states that "[y]ou can mount, access, and write to a data Storage Checkpoint just as you would to a file system." Furthermore, VxFS at p. 89 warns that such a writable checkpoint will diverge from a read-only storage checkpoint when it is written to. Changes made to a writable storage checkpoint are not reflected in the original file system. Once a VxFS storage checkpoint is mounted as an active file system, a storage snapshot can be taken of it.
36. A memory as in claim 35, wherein when changes are made to the first active file system or the second active file system, modified data is recorded in a location that is not shared with the third active file system.	Because VxFS storage checkpoints use copy-on-write techniques, as disclosed at pp. 83-85 and elsewhere, it is inherent that changes to any newly mounted active file systems and changes to the "parent" file system are stored in different locations so as to not overwrite any live data.
37. A memory as in claim 29, wherein the instructions further comprise the steps of: making a new snapshot of the second active file system, the new snapshot initially sharing data with the second active file system; converting the new snapshot to a third active file system by making the new snapshot writable, with changes made to the first active file system or the second active file system not reflected in the third active file system.	VxFS storage checkpoints can be mounted as writable checkpoints using the –o rw or –o remount options. (*See* p. 86, p. 89.) When such a storage checkpoint is mounted as writable, it initially shares data with the original file system through copy-on-write techniques. Once a storage checkpoint is mounted as writable, it is inherent that it becomes an independent active file system. For instance, VxFS at p. 86 states that "[y]ou can mount, access, and write to a data Storage Checkpoint just as you would to a file system." Furthermore, VxFS at p. 89 warns that such a writable checkpoint will diverge from a read-only storage checkpoint when it is written to. Thus, changes made to a writable storage checkpoint are not reflected in the original file system.

	Once a VxFS storage checkpoint is mounted as an active file system, a storage snapshot can be taken of it.
38. A memory as in claim 37, wherein when changes are made to the first active file system or the second active file system, modified data is recorded in a location that is not shared with the third active file system.	Because VxFS storage checkpoints use copy-on-write techniques, as disclosed at pp. 83-85 and elsewhere, it is inherent that changes to any newly mounted active file systems and changes to the "parent" file system are stored in different locations so as to not overwrite any live data.
39. A storage system, comprising: at least one storage device; an interface to at least one computing device or network for receiving and sending information; and a controller that controls storage and retrieval of the information in the storage device, the controller operating under program control to maintain plural active file systems, wherein each of the active file systems initially access data shared with another of the active file systems, and wherein changes made to each of the active file systems are not reflected in the active file system with which the changed active file system shares the data.	VxFS discloses a file system. It is inherent that the file system is comprised of computer-executable instructions operable to maintain file systems and acting though a storage controller. It is also well known and understood in the art that data inherently can be sent and received over a network when operating under file system control, such as VxFS. Chapter 8 discloses storage checkpoints, a copy-on-write snapshot of the file system. VxFS storage checkpoints can be mounted as writable checkpoints using the –o rw or –o remount options. (*See* p. 86, p. 89.) When such a storage checkpoint is mounted as writable, it initially shares data with the original file system through copy-on-write techniques. For instance, at pp. 84-85, VxFS teaches that "[t]he Storage Checkpoint presents the exact image of the file system by finding the data from the primary fileset. . . . This is called the *copy-on-write* technique, which allows the Storage Checkpoint to preserve the image of the primary fileset when the Storage Checkpoint is taken." Once a storage checkpoint is mounted as writable, it is inherent that it becomes an independent active file system. For instance, VxFS at p. 86 states that "[y]ou can mount, access, and write to a data Storage Checkpoint just as you would to a file system." Furthermore, VxFS at p. 89 warns that such a writable checkpoint will diverge from a read-only storage checkpoint when it is written to. Changes made to a writable storage checkpoint are not reflected in the original file system.
40. A storage system as in claim 39, wherein when a second active file system is created based on a first active file system, the first active file system and the second active file system initially share	VxFS at pp. 83-85 and elsewhere discloses that storage checkpoints are made using copy-on-write techniques. When such a storage checkpoint is mounted as writable, it initially shares data with the original file system.

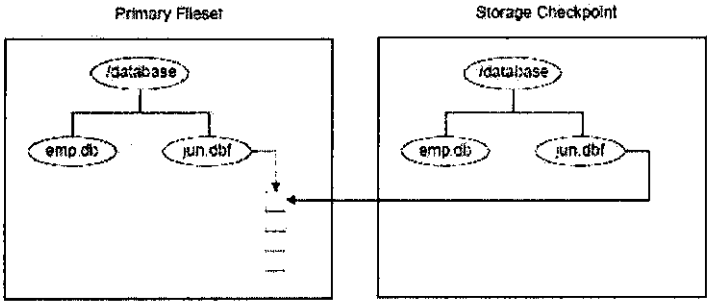
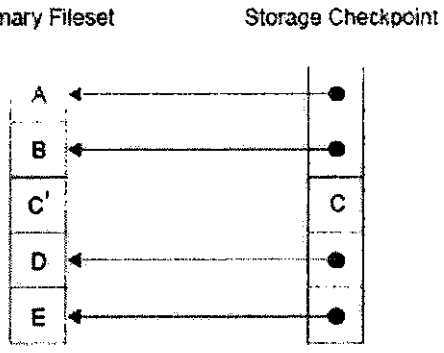
data.	
41. A storage system as in claim 40, wherein when changes are made to the first active file system, modified data is recorded in the first active file system in a location that is not shared with the second active file system.	Because VxFS storage checkpoints use copy-on-write techniques, as disclosed at pp. 83-85 and elsewhere, it is inherent that changes to either the writable storage checkpoint or the original file system are stored in different locations so as to not overwrite any live data.
42. A storage system as in claim 40, wherein when changes are made to the second active file system, modified data is recorded in the second active file system in a location that is not shared with the first active file system.	Because VxFS storage checkpoints use copy-on-write techniques, as disclosed at pp. 83-85 and elsewhere, it is inherent that changes to either the writable storage checkpoint or the original file system are stored in different locations so as to not overwrite any live data.
43. A storage system as in claim 39, wherein snapshots are made of ones of the plural active file systems, each snapshot forming an image of its respective active file system at a past consistency point.	VxFS at p. 83 teaches that storage checkpoints can be taken of multiple file systems. Thus, at p. 83 it is disclosed that [y]ou can create a Storage Checkpoint on a single file system or a list of file systems. A multiple file system Storage Checkpoint simultaneously freezes the file systems, creates a Storage Checkpoint on all file systems, and thaws the file systems." As disclosed at pp. 82-85, each checkpoint is an image of its file system at a past consistency point. Thus, at p. 83 it is disclosed that "[t]he storage checkpoint is logically identical to the primary fileset when the storage checkpoint is created." At pp. 84-85, VxFS teaches that "[t]he Storage Checkpoint presents the exact image of the file system by finding the data from the primary fileset. . . . This is called the *copy-on-write* technique, which allows the Storage Checkpoint to preserve the image of the primary fileset when the Storage Checkpoint is taken."
44. A storage system as in claim 43, wherein each snapshot includes a complete hierarchy for file system data, separate and apart from active file system data for the plural active file systems.	VxFS discloses on p. 86 and elsewhere that each storage snapshot contains file system metadata. For instance, as disclosed at pp. 83-85, this metadata includes pointers to data blocks. Appendix C further discloses that VxFS metadata includes a superblock, inodes and pointer tables for each file system.
45. A storage system as in claim 43, wherein at least one of the snapshots is converted into a new	VxFS discloses at pp. 83-85 and 89 that checkpoints can be taken of an active file system and that storage checkpoints can be mounted as writable, becoming other active file

active file system.	systems.
46. A storage system as in claim 45, wherein the one of the snapshots is converted by making the one of the snapshots writable.	VxFS discloses at pp. 83-85 and 89 that checkpoints can be taken of an active file system and that storage checkpoints can be mounted as writable, becoming other active file systems.
47. A storage system as in claim 46, wherein snapshot pointers from any of the active file systems to the new active file system are severed.	Once a storage checkpoint is mounted as writable, it is inherent that it becomes an independent active file system. For instance, VxFS at p. 86 states that "[y]ou can mount, access, and write to a data Storage Checkpoint just as you would to a file system." Furthermore, VxFS at p. 89 warns that such a writable checkpoint will diverge from a read-only storage checkpoint when it is written to. Therefore, no pointer exists between active file systems in VxFS.
48. A storage system, comprising: at least one storage device; an interface to at least one computing device or network for receiving and sending information; and a controller that controls storage and retrieval of the information in the storage device, the controller operating under program control to create plural active file systems, the program control comprising the steps of: making a snapshot of a first active file system, the snapshot initially sharing data with the first active file system; and converting the snapshot to a second active file system by making the snapshot writable, with changes made to the first active file system not reflected in the second active file system, and with changes made to the second active file system not reflected in the first active file system.	VxFS discloses a file system. It is inherent that the file system is comprised of computer-executable instructions operable to maintain file systems and acting though a storage controller. It is also well known and understood in the art that data inherently can be sent and received over a network when operating under file system control, such as VxFS. Chapter 8 discloses storage checkpoints, a copy-on-write snapshot of the file system. VxFS storage checkpoints can be mounted as writable checkpoints using the –o rw or –o remount options. (*See* p. 86, p. 89.) When such a storage checkpoint is mounted as writable, it initially shares data with the original file system through copy-on-write techniques. For instance, at pp. 84-85, VxFS teaches that "[t]he Storage Checkpoint presents the exact image of the file system by finding the data from the primary fileset. . . . This is called the *copy-on-write* technique, which allows the Storage Checkpoint to preserve the image of the primary fileset when the Storage Checkpoint is taken." Once a storage checkpoint is mounted as writable, it is inherent that it becomes an independent active file system. For instance, VxFS at p. 86 states that "[y]ou can mount, access, and write to a data Storage Checkpoint just as you would to a file system." Furthermore, VxFS at p. 89 warns that such a writable checkpoint will diverge from a read-only storage checkpoint when it is written to. Therefore, changes made to a writable storage checkpoint are not reflected in the original file system.
49. A storage system as in claim 48, wherein when changes are	Because VxFS storage checkpoints use copy-on-write techniques, as disclosed at pp. 83-85 and elsewhere, it is

made to the first active file system, modified data is recorded in the first active file system in a location that is not shared with the second active file system.	inherent that changes to either the writable storage checkpoint or the original file system are stored in different locations so as to not overwrite any live data.
50. A storage system as in claim 48, wherein when changes are made to the second active file system, modified data is recorded in the second active file system in a location that is not shared with the first active file system.	Because VxFS storage checkpoints use copy-on-write techniques, as disclosed at pp. 83-85 and elsewhere, it is inherent that changes to either the writable storage checkpoint or the original file system are stored in different locations so as to not overwrite any live data.
51. A storage system as in claim 48, wherein the program control further comprises the step of severing any snapshot pointers from the first active file system to the second active file system.	Once a storage checkpoint is mounted as writable, it is inherent that it becomes an independent active file system. For instance, VxFS at p. 86 states that "[y]ou can mount, access, and write to a data Storage Checkpoint just as you would to a file system." Furthermore, VxFS at p. 89 warns that such a writable checkpoint will diverge from a read-only storage checkpoint when it is written to. Therefore, no pointer exists between active file systems in VxFS.
52. A storage system as in claim 48, wherein the program control further comprises the steps of making snapshots of ones of the plural active file systems.	VxFS at p. 83 teaches that storage checkpoints can be taken of multiple file systems. Thus, at p. 83 it is disclosed that [y]ou can create a Storage Checkpoint on a single file system or a list of file systems. A multiple file system Storage Checkpoint simultaneously freezes the file systems, creates a Storage Checkpoint on all file systems, and thaws the file systems." At pp. 84-85, VxFS teaches that "[t]he Storage Checkpoint presents the exact image of the file system by finding the data from the primary fileset. . . . This is called the *copy-on-write* technique, which allows the Storage Checkpoint to preserve the image of the primary fileset when the Storage Checkpoint is taken."
53. A storage system as in claim 52, wherein each snapshot includes a complete hierarchy for file system data, separate and apart from active file system data for the plural active file systems.	VxFS discloses on p. 86 and elsewhere that each storage snapshot contains file system metadata. For instance, as disclosed at pp. 83-85, this metadata includes pointers to data blocks. Appendix C further discloses that VxFS metadata includes a superblock, inodes and pointer tables for each file system. The VxFS metadata is unique to each snapshot and to each active file system.
54. A storage system as in claim 48, wherein the program control	VxFS storage checkpoints can be mounted as writable checkpoints using the –o rw or –o remount options. (*See* p.

further comprises the steps of: making a new snapshot of the first active file system, the new snapshot initially sharing data with the first active file system; converting the new snapshot to a third active file system by making the new snapshot writable, with changes made to the first active file system or the second active file system not reflected in the third active file system.	86, p. 89.) When such a storage checkpoint is mounted as writable, it initially shares data with the original file system through copy-on-write techniques. Once a storage checkpoint is mounted as writable, it is inherent that it becomes an independent active file system. For instance, VxFS at p. 86 states that "[y]ou can mount, access, and write to a data Storage Checkpoint just as you would to a file system." Furthermore, VxFS at p. 89 warns that such a writable checkpoint will diverge from a read-only storage checkpoint when it is written to. Thus, changes made to a writable storage checkpoint are not reflected in the original file system. Once a VxFS storage checkpoint is mounted as an active file system, a storage snapshot can be taken of it.
55. A storage system as in claim 54, wherein when changes are made to the first active file system or the second active file system, modified data is recorded in a location that is not shared with the third active file system.	Because VxFS storage checkpoints use copy-on-write techniques, as disclosed at pp. 83-85 and elsewhere, it is inherent that changes to any newly mounted active file systems and changes to the "parent" file system are stored in different locations so as to not overwrite any live data.
56. A storage system as in claim 48, wherein the program control further comprises the steps of: making a new snapshot of the second active file system, the new snapshot initially sharing data with the second active file system; converting the new snapshot to a third active file system by making the new snapshot writable, with changes made to the first active file system or the second active file system not reflected in the third active file system.	VxFS storage checkpoints can be mounted as writable checkpoints using the –o rw or –o remount options. (*See* p. 86, p. 89.) When such a storage checkpoint is mounted as writable, it initially shares data with the original file system through copy-on-write techniques. Once a storage checkpoint is mounted as writable, it is inherent that it becomes an independent active file system. For instance, VxFS at p. 86 states that "[y]ou can mount, access, and write to a data Storage Checkpoint just as you would to a file system." Furthermore, VxFS at p. 89 warns that such a writable checkpoint will diverge from a read-only storage checkpoint when it is written to. Changes made to a writable storage checkpoint are not reflected in the original file system. Once a VxFS storage checkpoint is mounted as an active file system, a storage snapshot can be taken of it.
57. A storage system as in claim 56, wherein when changes are	Because VxFS storage checkpoints use copy-on-write techniques, as disclosed at pp. 83-85 and elsewhere, it is

made to the first active file system or the second active file system, modified data is recorded in a location that is not shared with the third active file system.	inherent that changes to any newly mounted active file systems and changes to the "parent" file system are stored in different locations so as to not overwrite any live data.
58. An apparatus for operating data storage, the apparatus including means for creating plural active file systems and means for maintaining plural active file systems, wherein each of the active file systems initially access data shared with another of the active file systems, and wherein changes made to each of the active file systems are not reflected in the active file system with which the changed active file system shares the data.	VxFS discloses a file system. Chapter 8 discloses storage checkpoints, a copy-on-write snapshot of the file system. VxFS storage checkpoints can be mounted as writable checkpoints using the –o rw or –o remount options. (*See* p. 86, p. 89.) When such a storage checkpoint is mounted as writable, it initially shares data with the original file system through copy-on-write techniques. For instance, at pp. 84-85, VxFS teaches that "[t]he Storage Checkpoint presents the exact image of the file system by finding the data from the primary fileset. . . . This is called the *copy-on-write* technique, which allows the Storage Checkpoint to preserve the image of the primary fileset when the Storage Checkpoint is taken." Once a storage checkpoint is mounted as writable, it is inherent that it becomes an independent active file system. For instance, VxFS at p. 86 states that "[y]ou can mount, access, and write to a data Storage Checkpoint just as you would to a file system." Furthermore, VxFS at p. 89 warns that such a writable checkpoint will diverge from a read-only storage checkpoint when it is written to. Changes made to a writable storage checkpoint are not reflected in the original file system.
59. An apparatus for creating plural active file systems, comprising: means for making a snapshot of a first active file system, the snapshot initially sharing data with the first active file system; and means for converting the snapshot to a second active file system by making the snapshot writable, with changes made to the first active file system not reflected in the second active file system, and with changes made to the second active file system not reflected in the first active file system.	VxFS discloses a file system. Chapter 8 discloses storage checkpoints, a copy-on-write snapshot of the file system. VxFS storage checkpoints can be mounted as writable checkpoints using the –o rw or –o remount options. (*See* p. 86, p. 89.) When such a storage checkpoint is mounted as writable, it initially shares data with the original file system through copy-on-write techniques. For instance, at pp. 84-85, VxFS teaches that "[t]he Storage Checkpoint presents the exact image of the file system by finding the data from the primary fileset. . . . This is called the *copy-on-write* technique, which allows the Storage Checkpoint to preserve the image of the primary fileset when the Storage Checkpoint is taken." Once a storage checkpoint is mounted as writable, it is inherent that it becomes an independent active file system. For instance, VxFS at p. 86 states that "[y]ou can mount, access, and write to a data Storage Checkpoint just as you

	would to a file system." Furthermore, VxFS at p. 89 warns that such a writable checkpoint will diverge from a read-only storage checkpoint when it is written to. Changes made to a writable storage checkpoint are not reflected in the original file system.
60. A method of operating data storage, comprising: making a snapshot of organizational data of a first active file system, the snapshot pointing to original non-organizational data of the first active file system; storing the snapshot; modifying a first portion of the original non-organizational data of the first active file system in response to a first active file system access request, resulting in a modified first portion being part of first modified non-organizational data of the first active file system; and storing the modified first portion so as not to overwrite the first portion; wherein, after the step of storing the modified first portion, the snapshot points to the original non-organizational data, the organizational data of the first active file system point to the first modified non-organizational data of the first filing system, and the original non-organizational data and the first modified non-organizational data partially overlap.	VxFS discloses at pp. 83-85 making a snapshot of organizational data (file system metadata) of a first file system, the snapshot pointing to original non-organizational data (data blocks): Figure 2. Primary Fileset and its Storage Checkpoint  VxFS further teaches at pp. 83-85 that as data blocks are modified, they are stored using copy-on-write so as not to overwrite the old data, which is copied to a new location. After data blocks are updated, the checkpoint points to the original data, which partially overlaps with the changed data blocks: Figure 4. Updates to the Primary Fileset  VxFS further discloses at pp. 83-85 that the active file system uses a block map to point to updated data blocks.
61. A method according to claim 60, wherein: the step of storing the snapshot comprises storing the	VxFS storage checkpoints can be mounted as writable checkpoints using the –o rw or –o remount options. (*See* p. 86, p. 89.) When such a storage checkpoint is mounted as

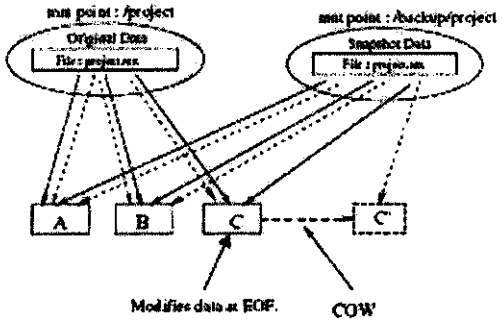
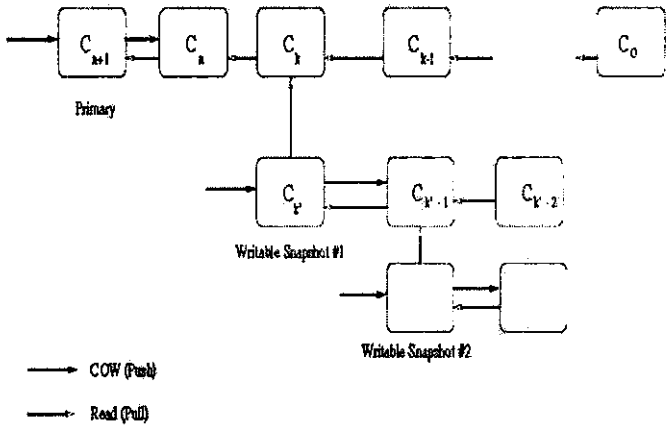
snapshot as a second active filing system; the method further comprising: modifying a second portion of the original non-organizational data in response to a second active file system access request, resulting in a modified second portion being part of second modified non-organizational data of the second active file system; and storing the modified second portion so as not to overwrite the second portion; wherein, after the step of storing the modified second portion, the snapshot points to the second modified non-organizational data, the organizational data of the first active file system point to the first modified non-organizational data, and the first modified non-organizational data and the second modified non-organizational data partially overlap.	writable, it initially shares data with the original file system through copy-on-write techniques, so as not to overwrite any data. For instance, at pp. 84-85, VxFS teaches that "[t]he Storage Checkpoint presents the exact image of the file system by finding the data from the primary fileset. . . . This is called the *copy-on-write* technique, which allows the Storage Checkpoint to preserve the image of the primary fileset when the Storage Checkpoint is taken." It is inherent that if a VxFS storage checkpoint is mounted as an active file system, its metadata will point to the modified data blocks that are written to it, as well as to unmodified data blocks that it still shares with the parent active file system
62. A method according to claim 61, wherein the step of making a snapshot is performed at a consistency point of the first active file system.	As disclosed at pp. 82-85, each checkpoint is an image of its file system at a past consistency point. Thus, at p. 83 it is disclosed that "[t]he storage checkpoint is logically identical to the primary fileset when the storage checkpoint is created." At pp. 84-85, VxFS teaches that "[t]he Storage Checkpoint presents the exact image of the file system by finding the data from the primary fileset. . . . This is called the *copy-on-write* technique, which allows the Storage Checkpoint to preserve the image of the primary fileset when the Storage Checkpoint is taken."
63. A method according to claim 62, wherein data is stored in the first and second active file systems using blocks.	VxFS at e.g. pp. 83-85 discloses that data is stored using data blocks.

Third Basis of Invalidity

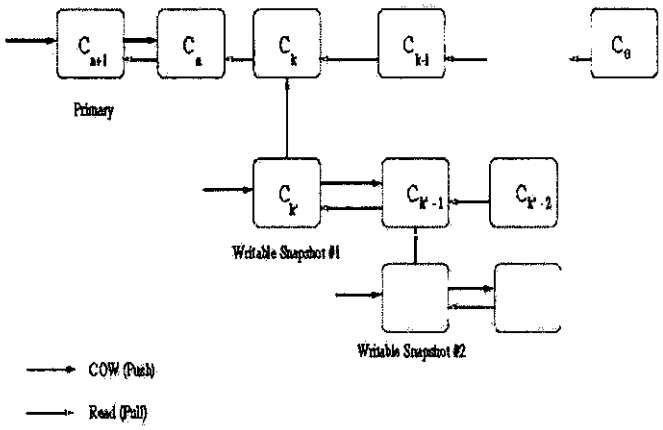
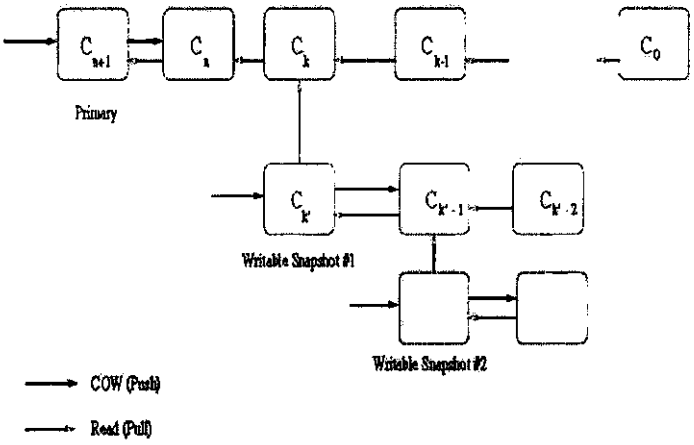
The reference applicable to the third basis of invalidity is:

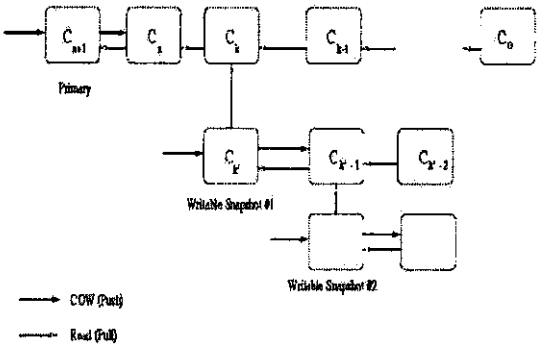
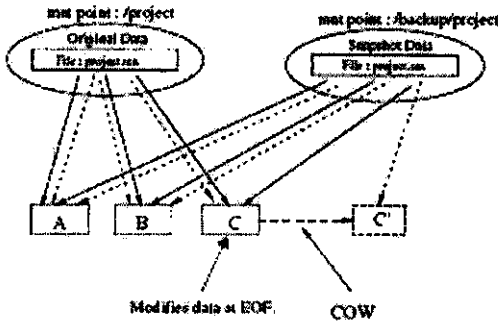
1. S. B. Siddha, K. Gopinath, *A Persistent Snapshot Device Driver for Linux*, Proceedings of the 5th Annual Linux Showcase & Conference, USENIX, Nov. 5-10, 2001. ("Siddha").

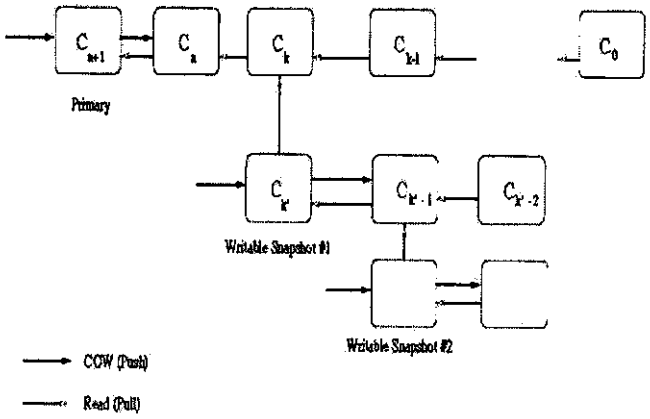
The pertinence and manner of applying Siddha to claims 1-63 for which re-examination is requested is as follows:

1. A method of operating data storage, the method including maintenance of plural active file systems, wherein each of the active file systems initially access data shared with another of the active file systems, and wherein changes made to each of the active file systems are not reflected in the active file system with which the changed active file system shares the data.	Siddha generally and section 3.1.3 specifically teaches a file system with a multiple snapshot capability, wherein the snapshots are established using copy-on-write techniques so as not to overwrite data, as shown in Fig. 1:  Figure 1: Snapshot COW Mechanism Siddha section 3.1.3 further teaches that snapshots can be made writable, establishing a diverging snapshot tree of multiple active file systems, as depicted in Fig. 4:  Figure 4: Snapshot Tree Snapshot Ck' is even labeled as "Writable Snapshot #1." Siddha teaches that, because of the copy-on-write nature of the snapshots, active file systems formed by mounting snapshots as writable initially share data sets with parent file systems and diverge over time. Thus, in Fig. 4, writable snapshot Ck' initially shares data with snapshot Ck, which represents the state of the

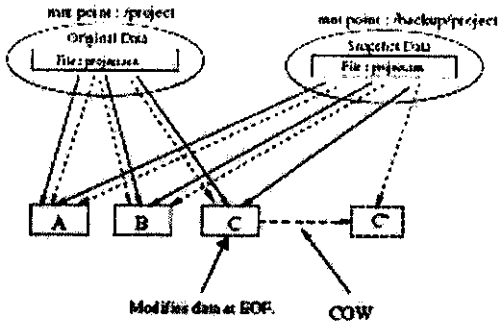
	original file system at time k. Snapshot Ck'-1, representing the state of the new file system at a later time, does not share changed data with snapshot Ck-1, which represents the state of the original file system.
2. A method as in claim 1, wherein when a second active file system is created based on a first active file system, the first active file system and the second active file system initially share data.	Siddha teaches that, because of the copy-on-write nature of the snapshots, active file systems formed by mounting snapshots as writable initially share data sets with parent file systems and diverge over time. Thus, in Fig. 4, writable snapshot Ck' initially shares data with snapshot Ck, which represents the state of the original file system at time k. Snapshot Ck'-1, representing the state of the new file system at a later time, does not share changed data with snapshot Ck-1, which represents the state of the original file system.
3. A method as in claim 2, wherein when changes are made to the first active file system, modified data is recorded in the first active file system in a location that is not shared with the second active file system.	Due to the copy-on-write nature of snapshots disclosed in Siddha, modified data in different active file systems is inherently written in new locations so as to not overwrite other data, as shown in Fig. 1.
4. A method as in claim 2, wherein when changes are made to the second active file system, modified data is recorded in the second active file system in a location that is not shared with the first active file system.	Due to the copy-on-write nature of snapshots disclosed in Siddha, modified data in different active file systems is inherently written in new locations so as to not overwrite other data, as shown in Fig. 1.
5. A method as in claim 1, wherein snapshots are made of ones of the plural active file systems, each snapshot forming an image of its respective active file system at a past consistency point.	Siddha in section 3.1.3 and Fig. 4 teaches snapshots of plural active file systems. Section 3.1.3 further teaches the need to keep snapshots consistent through atomic updates. Fig. 4 shows each snapshot to be a past consistent view of active file systems. Section 3.1.3 further teaches that "[w]e can have again read-only snapshots from read-write snapshots and from these read-only snapshots we can again have read-write snapshots. Thus we have a tree like structure as shown in Fig. 4."

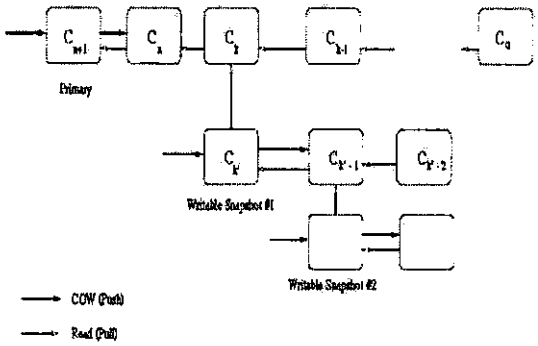
	 Figure 4: Snapshot Tree
6. A method as in claim 5, wherein each snapshot includes a complete hierarchy for file system data, separate and apart from active file system data for the plural active file systems.	Sections 1.1, 3.1.3 and Figs. 1 and 4 teach that each snapshot has a hierarchy of metadata that is unique to that snapshot. This is further shown in Fig. 5 – each snapshot has its own map. It is further inherent in the copy-on-write technique that each snapshot and each active file system have unique metadata (such as block maps).
7. A method as in claim 5, wherein at least one of the snapshots is converted into a new active file system.	Siddha section 3.1.3 teaches that "[w]e can have again read-only snapshots from read-write snapshots and from these read-only snapshots we can again have read-write snapshots. Thus we have a tree like structure as shown in Fig. 4."  Figure 4: Snapshot Tree
8. A method as in claim 7, wherein the one of the snapshots is converted by	Siddha section 3.1.3 teaches that "[w]e can have again read-only snapshots from read-write snapshots and from these read-only snapshots we can again have read-write snapshots. Thus we have

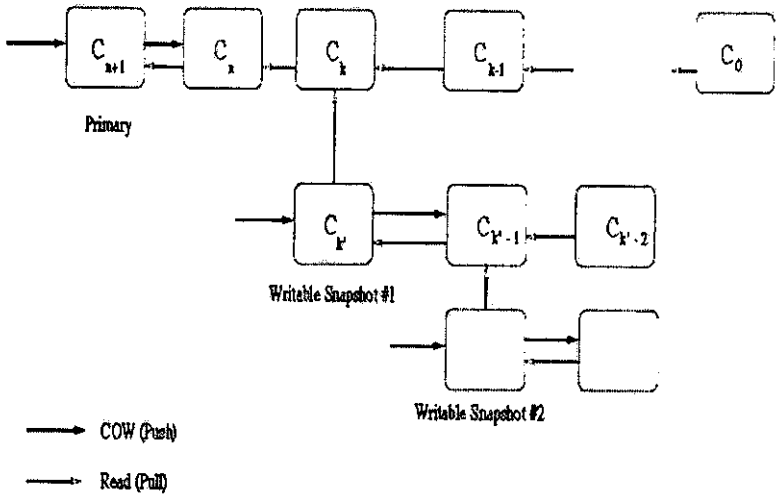
making the one of the snapshots writable.	a tree like structure as shown in Fig. 4."  Figure 4: Snapshot Tree
9. A method as in claim 8, wherein snapshot pointers from any of the active file systems to the new active file system are severed.	Siddha Fig. 4 teaches that there are no pointers between any of the active file systems. Section 3.1.3 further teaches that "[t]he basic idea is to associate an additional snapshot to every snapshot that needs to be mounted writable. All actual writes are then made to this new snapshot and the snapshot chain is modified such that no downstream snapshot references this one."
10. A method of creating plural active file systems, comprising the steps of: making a snapshot of a first active file system, the snapshot initially sharing data with the first active file system; and converting the snapshot to a second active file system by making the snapshot writable, with changes made to the first active file system not reflected in the second active file system, and with changes made to the second active file system not reflected in the first active file system.	Siddha generally and section 3.1.3 specifically teaches a file system with a multiple snapshot capability, wherein the snapshots are established using copy-on-write techniques so as not to overwrite data, as shown in Fig. 1:  Figure 1: Snapshot COW Mechanism Siddha section 3.1.3 further teaches that snapshots can be made writable, establishing a diverging snapshot tree of multiple active file systems, as depicted in Fig. 4:

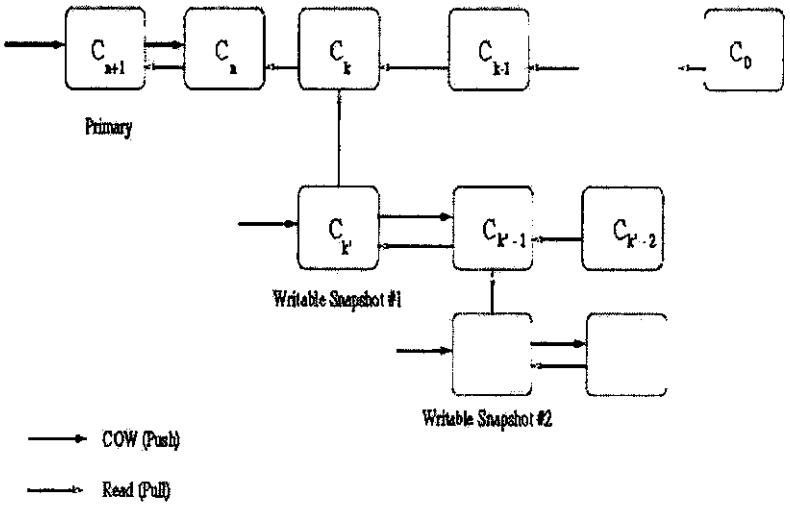
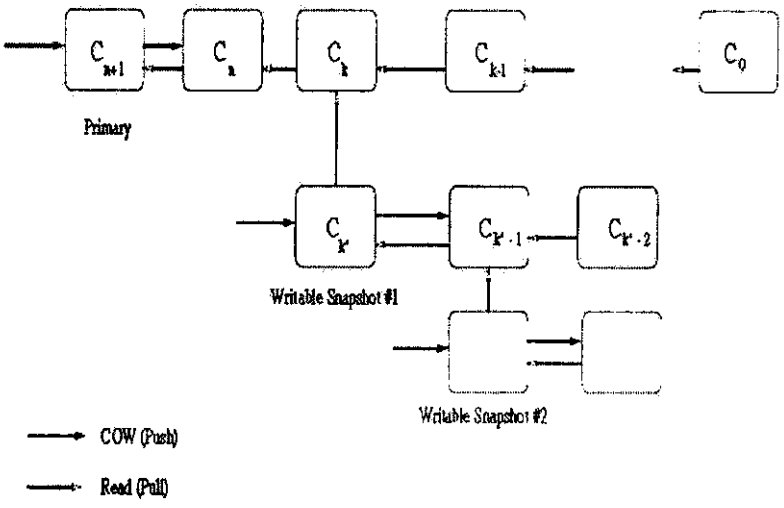
	 Figure 4: Snapshot Tree Siddha teaches that, because of the copy-on-write nature of the snapshots, active file systems formed by mounting snapshots as writable initially share data sets with parent file systems and diverge over time. Thus, in Fig. 4, writable snapshot Ck' initially shares data with snapshot Ck, which represents the state of the original file system at time k. Snapshot Ck'-1, representing the state of the new file system at a later time, does not share changed data with snapshot Ck-1, which represents the state of the original file system.
11. A method as in claim 10, wherein when changes are made to the first active file system, modified data is recorded in the first active file system in a location that is not shared with the second active file system.	Due to the copy-on-write nature of snapshots disclosed in Siddha, modified data in different active file systems is inherently written in new locations so as to not overwrite other data, as shown in Fig. 1.
12. A method as in claim 10, wherein when changes are made to the second active file system, modified data is recorded in the second active file system in a location that is not shared with the first active file system.	Due to the copy-on-write nature of snapshots disclosed in Siddha, modified data in different active file systems is inherently written in new locations so as to not overwrite other data, as shown in Fig. 1.
13. A method as in claim 10, further comprising the step of severing any snapshot pointers from the first active	Siddha Fig. 4 teaches that there are no pointers between any of the active file systems. Section 3.1.3 further teaches that "[t]he basic idea is to associate an additional snapshot to every snapshot that needs to be mounted writable. All actual writes are then

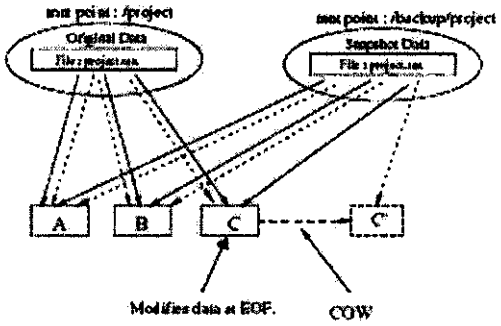
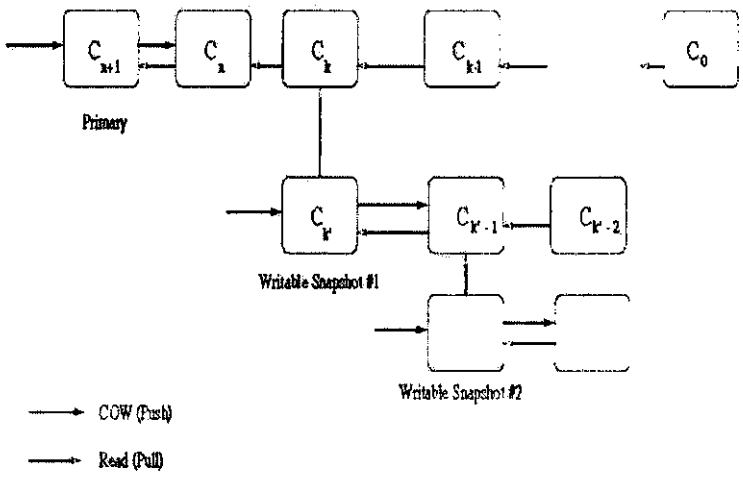
file system to the second active file system.	made to this new snapshot and the snapshot chain is modified such that no downstream snapshot references this one."
14. A method as in claim 10, further comprising the steps of making snapshots of ones of the plural active file systems.	Siddha in section 3.1.3 and Fig. 4 teaches snapshots of plural active file systems Section 3.1.3 further teaches that "[w]e can have again read-only snapshots from read-write snapshots and from these read-only snapshots we can again have read-write snapshots. Thus we have a tree like structure as shown in Fig. 4."
15. A method as in claim 14, wherein each snapshot includes a complete hierarchy for file system data, separate and apart from active file system data for the plural active file systems.	Sections 1.1, 3.1.3 and Figs. 1 and 4 teach that each snapshot has a hierarchy of metadata that is unique to that snapshot. This is further shown in Fig. 5 – each snapshot has its own map. It is further inherent in the copy-on-write technique that each snapshot and each active file system have unique metadata (such as block maps).
16. A method as in claim 10, further comprising the steps of: making anew snapshot of the first active file system, the new snapshot initially sharing data with the first active file system; converting the new snapshot to a third active file system by making the new snapshot writable, with changes made to the first active file system or the second active file system not reflected in the third active file system.	Siddha in section 3.1.3 and Fig. 4 teaches snapshots of plural active file systems Section 3.1.3 further teaches that "[w]e can have again read-only snapshots from read-write snapshots and from these read-only snapshots we can again have read-write snapshots. Thus we have a tree like structure as shown in Fig. 4." Siddha teaches that, because of the copy-on-write nature of the snapshots, active file systems formed by mounting snapshots as writable initially share data sets with parent file systems and diverge over time. Changes to any of the active file systems are not reflected in the other active file systems, as shown in Fig. 4 and taught in section 3.1.3: "The basic idea is to associate an additional snapshot to every snapshot that needs to be mounted writable. All actual writes are then made to this new snapshot and the snapshot chain is modified such that no downstream snapshot references this one."
17. A method as in claim 16, wherein when changes are made to the first active file system or the second active file system, modified data is recorded in a location that is not shared with the third active file system.	Due to the copy-on-write nature of snapshots disclosed in Siddha, modified data in different active file systems is inherently written in new locations so as to not overwrite other data, as shown in Fig. 1.
18. A method as in claim 10, further comprising the steps	Siddha in section 3.1.3 and Fig. 4 teaches snapshots of plural active file systems Section 3.1.3 further teaches that "[w]e can

of: making a new snapshot of the second active file system, the new snapshot initially sharing data with the second active file system; converting the new snapshot to a third active file system by making the new snapshot writable, with changes made to the first active file system or the second active file system not reflected in the third active file system.	have again read-only snapshots from read-write snapshots and from these read-only snapshots we can again have read-write snapshots. Thus we have a tree like structure as shown in Fig. 4." Siddha teaches that, because of the copy-on-write nature of the snapshots, active file systems formed by mounting snapshots as writable initially share data sets with parent file systems and diverge over time. Changes to any of the active file systems are not reflected in the other active file systems, as shown in Fig. 4 and taught in section 3.1.3: "The basic idea is to associate an additional snapshot to every snapshot that needs to be mounted writable. All actual writes are then made to this new snapshot and the snapshot chain is modified such that no downstream snapshot references this one."
19. A method as in claim 18, wherein when changes are made to the first active file system or the second active file system, modified data is recorded in a location that is not shared with the third active file system.	Due to the copy-on-write nature of snapshots disclosed in Siddha, modified data in different active file systems is inherently written in new locations so as to not overwrite other data, as shown in Fig. 1.
20. A memory storing information including instructions, the instructions executable by a processor to operate data storage, the instructions comprising steps to maintain plural active file systems, wherein each of the active file systems initially access data shared with another of the active file systems, and wherein changes made to each of the active file systems are not reflected in the active file system with which the changed active file system shares the data.	It is inherent that the file system in Siddha is in the form of computer-executable instructions stored in a memory. Siddha generally and section 3.1.3 specifically teaches a file system with a multiple snapshot capability, wherein the snapshots are established using copy-on-write techniques so as not to overwrite data, as shown in Fig. 1:  Figure 1: Snapshot COW Mechanism Siddha section 3.1.3 further teaches that snapshots can be made writable, establishing a diverging snapshot tree of multiple active file systems, as depicted in Fig. 4:

	 Figure 4: Snapshot Tree Siddha teaches that, because of the copy-on-write nature of the snapshots, active file systems formed by mounting snapshots as writable initially share data sets with parent file systems and diverge over time. Thus, in Fig. 4, writable snapshot Ck' initially shares data with snapshot Ck, which represents the state of the original file system at time k. Snapshot Ck'-1, representing the state of the new file system at a later time, does not share changed data with snapshot Ck-1, which represents the state of the original file system.
21. A memory as in claim 20, wherein when a second active file system is created based on a first active file system, the first active file system and the second active file system initially share data.	Siddha teaches that, because of the copy-on-write nature of the snapshots, active file systems formed by mounting snapshots as writable initially share data sets with parent file systems and diverge over time. Thus, in Fig. 4, writable snapshot Ck' initially shares data with snapshot Ck, which represents the state of the original file system at time k. Snapshot Ck'-1, representing the state of the new file system at a later time, does not share changed data with snapshot Ck-1, which represents the state of the original file system.
22. A memory as in claim 21, wherein when changes are made to the first active file system, modified data is recorded in the first active file system in a location that is not shared with the second active file system.	Due to the copy-on-write nature of snapshots disclosed in Siddha, modified data in different active file systems is inherently written in new locations so as to not overwrite other data, as shown in Fig. 1.
23. A memory as in claim 21, wherein when changes are made to the second active file system, modified data is recorded in the second active file system in	Due to the copy-on-write nature of snapshots disclosed in Siddha, modified data in different active file systems is inherently written in new locations so as to not overwrite other data, as shown in Fig. 1.

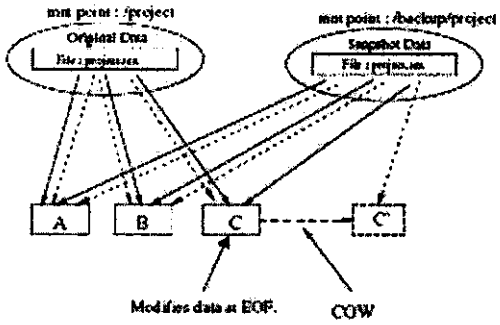
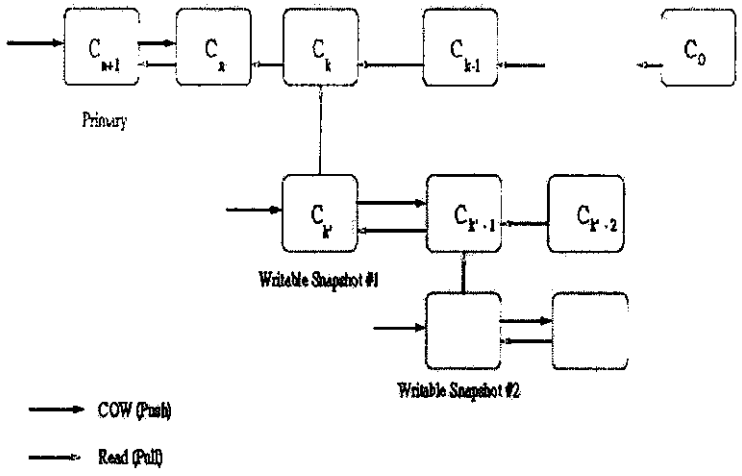
a location that is not shared with the first active file system.	
24. A memory as in claim 20, wherein snapshots are made of ones of the plural active file systems, each snapshot forming an image of its respective active file system at a past consistency point.	Siddha in section 3.1.3 and Fig. 4 teaches snapshots of plural active file systems. Section 3.1.3 further teaches the need to keep snapshots consistent through atomic updates. Fig. 4 shows each snapshot to be a past consistent view of active file systems. Section 3.1.3 further teaches that "[w]e can have again read-only snapshots from read-write snapshots and from these read-only snapshots we can again have read-write snapshots. Thus we have a tree like structure as shown in Fig. 4."  Figure 4: Snapshot Tree
25. A method as in claim 24, wherein each snapshot includes a complete hierarchy for file system data, separate and apart from active file system data for the plural active file systems.	Sections 1.1, 3.1.3 and Figs. 1 and 4 teach that each snapshot has a hierarchy of metadata that is unique to that snapshot. This is further shown in Fig. 5 – each snapshot has its own map. It is further inherent in the copy-on-write technique that each snapshot and each active file system have unique metadata (such as block maps).
26. A memory as in claim 24, wherein at least one of the snapshots is converted into a new active file system.	Siddha section 3.1.3 teaches that "[w]e can have again read-only snapshots from read-write snapshots and from these read-only snapshots we can again have read-write snapshots. Thus we have a tree like structure as shown in Fig. 4."

	 Figure 4: Snapshot Tree
27. A memory as in claim 26, wherein the one of the snapshots is converted by making the one of the snapshots writable.	Siddha section 3.1.3 teaches that "[w]e can have again read-only snapshots from read-write snapshots and from these read-only snapshots we can again have read-write snapshots. Thus we have a tree like structure as shown in Fig. 4."  Figure 4: Snapshot Tree
28. A memory as in claim 27, wherein snapshot pointers from any of the active file systems to the new active file system are severed.	Siddha Fig. 4 teaches that there are no pointers between any of the active file systems. Section 3.1.3 further teaches that "[t]he basic idea is to associate an additional snapshot to every snapshot that needs to be mounted writable. All actual writes are then made to this new snapshot and the snapshot chain is modified such that no downstream snapshot references this one."

29. A memory storing information including instructions, the instructions executable by a processor to create plural active file systems, the instructions comprising the steps of: making a snapshot of a first active file system, the snapshot initially sharing data with the first active file system; and converting the snapshot to a second active file system by making the snapshot writable, with changes made to the first active file system not reflected in the second active file system, and with changes made to the second active file system not reflected in the first active file system.	It is inherent that the file system in Siddha is in the form of computer-executable instructions stored in a memory. Siddha generally and section 3.1.3 specifically teaches a file system with a multiple snapshot capability, wherein the snapshots are established using copy-on-write techniques so as not to overwrite data, as shown in Fig. 1:  Figure 1: Snapshot COW Mechanism Siddha section 3.1.3 further teaches that snapshots can be made writable, establishing a diverging snapshot tree of multiple active file systems, as depicted in Fig. 4:  Figure 4: Snapshot Tree Siddha teaches that, because of the copy-on-write nature of the snapshots, active file systems formed by mounting snapshots as writable initially share data sets with parent file systems and diverge over time. Thus, in Fig. 4, writable snapshot Ck' initially shares data with snapshot Ck, which represents the state of the original file system at time k. Snapshot Ck'-1, representing the state of the new file system at a later time, does not share changed data with snapshot Ck-1, which represents the state of the original file system.

30. A memory as in claim 29, wherein when changes are made to the first active file system, modified data is recorded in the first active file system in a location that is not shared with the second active file system.	Due to the copy-on-write nature of snapshots disclosed in Siddha, modified data in different active file systems is inherently written in new locations so as to not overwrite other data, as shown in Fig. 1.
31. A memory as in claim 29, wherein when changes are made to the second active file system, modified data is recorded in the second active file system in a location that is not shared with the first active file system.	Due to the copy-on-write nature of snapshots disclosed in Siddha, modified data in different active file systems is inherently written in new locations so as to not overwrite other data, as shown in Fig. 1.
32. A memory as in claim 29, wherein the instructions further comprise the step of severing any snapshot pointers from the first active file system to the second active file system.	Siddha Fig. 4 teaches that there are no pointers between any of the active file systems. Section 3.1.3 further teaches that "[t]he basic idea is to associate an additional snapshot to every snapshot that needs to be mounted writable. All actual writes are then made to this new snapshot and the snapshot chain is modified such that no downstream snapshot references this one."
33. A memory as in claim 29, wherein the instructions further comprise the steps of making snapshots of ones of the plural active file systems.	Siddha in section 3.1.3 and Fig. 4 teaches snapshots of plural active file systems Section 3.1.3 further teaches that "[w]e can have again read-only snapshots from read-write snapshots and from these read-only snapshots we can again have read-write snapshots. Thus we have a tree like structure as shown in Fig. 4."
34. A memory as in claim 33, wherein each snapshot includes a complete hierarchy for file system data, separate and apart from active file system data for the plural active file systems.	Sections 1.1, 3.1.3 and Figs. 1 and 4 teach that each snapshot has a hierarchy of metadata that is unique to that snapshot. This is further shown in Fig. 5 – each snapshot has its own map. It is further inherent in the copy-on-write technique that each snapshot and each active file system have unique metadata (such as block maps).
35. A memory as in claim 29, wherein the instructions further comprise the steps	Siddha in section 3.1.3 and Fig. 4 teaches snapshots of plural active file systems Section 3.1.3 further teaches that "[w]e can have again read-only snapshots from read-write snapshots and

of: making a new snapshot of the first active file system, the new snapshot initially sharing data with the first active file system; converting the new snapshot to a third active file system by making the new snapshot writable, with changes made to the first active file system or the second active file system not reflected in the third active file system.	from these read-only snapshots we can again have read-write snapshots. Thus we have a tree like structure as shown in Fig. 4." Siddha teaches that, because of the copy-on-write nature of the snapshots, active file systems formed by mounting snapshots as writable initially share data sets with parent file systems and diverge over time. Changes to any of the active file systems are not reflected in the other active file systems, as shown in Fig. 4 and taught in section 3.1.3: "The basic idea is to associate an additional snapshot to every snapshot that needs to be mounted writable. All actual writes are then made to this new snapshot and the snapshot chain is modified such that no downstream snapshot references this one."
36. A memory as in claim 35, wherein when changes are made to the first active file system or the second active file system, modified data is recorded in a location that is not shared with the third active file system.	Due to the copy-on-write nature of snapshots disclosed in Siddha, modified data in different active file systems is inherently written in new locations so as to not overwrite other data, as shown in Fig. 1.
37. A memory as in claim 29, wherein the instructions further comprise the steps of: making a new snapshot of the second active file system, the new snapshot initially sharing data with the second active file system; converting the new snapshot to a third active file system by making the new snapshot writable, with changes made to the first active file system or the second active file system not reflected in the third active file system.	Siddha in section 3.1.3 and Fig. 4 teaches snapshots of plural active file systems Section 3.1.3 further teaches that "[w]e can have again read-only snapshots from read-write snapshots and from these read-only snapshots we can again have read-write snapshots. Thus we have a tree like structure as shown in Fig. 4." Siddha teaches that, because of the copy-on-write nature of the snapshots, active file systems formed by mounting snapshots as writable initially share data sets with parent file systems and diverge over time. Changes to any of the active file systems are not reflected in the other active file systems, as shown in Fig. 4 and taught in section 3.1.3: "The basic idea is to associate an additional snapshot to every snapshot that needs to be mounted writable. All actual writes are then made to this new snapshot and the snapshot chain is modified such that no downstream snapshot references this one."
38. A memory as in claim 37, wherein when changes are made to the first active file system or the second active file system, modified	Due to the copy-on-write nature of snapshots disclosed in Siddha, modified data in different active file systems is inherently written in new locations so as to not overwrite other data, as shown in Fig. 1.

data is recorded in a location that is not shared with the third active file system.	
39. A storage system, comprising: at least one storage device; an interface to at least one computing device or network for receiving and sending information; and a controller that controls storage and retrieval of the information in the storage device, the controller operating under program control to maintain plural active file systems, wherein each of the active file systems initially access data shared with another of the active file systems, and wherein changes made to each of the active file systems are not reflected in the active file system with which the changed active file system shares the data.	It is inherent that the file system in Siddha is executed in conjunction with a computer storage controller. Program and disk implementation of the Siddha file system is disclosed in section 4. One of skill in the art would readily appreciate that the Siddha file system is inherently implemented in conjunction with a computer and/or network interfaces. Siddha generally and section 3.1.3 specifically teaches a file system with a multiple snapshot capability, wherein the snapshots are established using copy-on-write techniques so as not to overwrite data, as shown in Fig. 1:  Figure 1: Snapshot COW Mechanism Siddha section 3.1.3 further teaches that snapshots can be made writable, establishing a diverging snapshot tree of multiple active file systems, as depicted in Fig. 4:  Figure 4: Snapshot Tree Siddha teaches that, because of the copy-on-write nature of the snapshots, active file systems formed by mounting snapshots as